МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет математики, информационных и авиационных технологий
Кафедра «Телекоммуникационных технологий и сетей»

***Чичев А.А., Чекал Е.Г.***

# СЕТЕВОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

*Учебное пособие*
*2-е издание, исправленное и дополненное*

Ульяновск
2020

**УДК 004.7**
**ББК 32.971**
**Ч-72**

*Печатается по решению Ученого совета*
*факультета математики, информационных и авиационных технологий*
*Ульяновского государственного университета*
*(протокол № 7/20 от 20.10.2020)*

**Рецензенты:**
заведующий кафедрой телекоммуникационных технологий и сетей УлГУ, д.т.н., профессор *А.А. Смагин;*
доцент кафедры «Прикладная математика и информатика» к.т.н., *Т.Е. Родионова*

**Чичев А.А.**
**Ч-72** **Сетевое программное обеспечение** : учебное пособие / А.А. Чичев , Е.Г. Чекал. - 2-е изд., испр. и доп. – Ульяновск : УлГУ, 2020. – 166 с.

Учебное пособие составлено в соответствии с программой дисциплины «Сетевое программное обеспечение», и предусматривает подготовку магистров, инженеров и бакалавров по направлениям 11.03.02 «Инфокоммуникационные технологии и системы связи», 02.03.03 «Математическое обеспечение и администрирование информационных систем», 09.03.02 «Информационные системы и технологии», 09.03.03 «Прикладная информатика» и специальности 10.05.01 «Компьютерная безопасность». Может использоваться студентами родственных специальностей и направлений.

В теоретической части пособия рассмотрены общие вопросы организации сетей, приведены сведения о структуре стеков сетевых протоколов, о наименовании взаимодействующих узлов в сети, о структуре Интернета, о сетевых сервисах и управлении ими в вычислительных системах. В практической части приводятся методические указания к некоторым лабораторным работам по установке, конфигурированию и эксплуатации сетевых сервисов.

Пособие предназначено для практического руководства при проведении преподавателями лекционных и лабораторных занятий со студентами указанных направлений и специальностей всех форм обучения, а также для самостоятельной работы студентов.

УДК 004.4'6(075.8)
ББК 32.972.1я73

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

**Исторически сложилось так**, что в настоящее время самой быстро развивающейся, самой функционально продвинутой, самой сложной алгоритмически, самой большой по объёму и вообще самой-самой операционной системой стала операционная система Linux. Возможно, кому-то это высказывание не нравится, но, увы, оно истинно.

Именно в развитие этой операционной системы вкладываются усилия тысяч высококвалифицированных разработчиков, как частных лиц, так и сотрудников фирм по всему миру. Объём проекта (Linux-5.x) оценивается в 20 (двадцать!) млн. строк кода — ещё никогда человечество не разрабатывало столь сложные и объёмные программные системы.

Именно эта операционная система обеспечивает функционирование самых мощных ЭВМ нашего мира — согласно списка Top-500 её используют более 95% суперЭВМ [5].

Именно над развитием этой операционной системы работают сотрудники крупнейших корпораций мира — IBM, HP, Oracle, Intel, Google, Samsung и других, и даже Microsoft (!). Ещё совсем недавно, в 90-ые годы, основной вклад в развитие Linux делали частные лица. Однако, в последние десятилетия в крупнейших ИТ-корпорациях мира появились подразделения, специализирующиеся на разработке в Linux и на её совершенствовании.

Именно эта операционная система совместно с другими Unix'ами является основой Интернета, а «Интернет — это наше всё».

Именно эта операционная система совместно с другими Unix'ами и Unix-подобными ОС является базой для построения высоконадёжных высокодоступных (24x7) информационных систем.

Именно эта операционная система наряду с другими Unix'ами используется на серверах корпораций, банков, бирж, является основой систем управления в энергетике (в том числе АЭС), на транспорте (диспетчерские системы), в связи (АТС, провайдеры Интернета и сотовой связи), в государственных структурах.

То есть, зависимость современной экономики от Unix'овых операционных систем (и прежде всего, Linux) не просто большая, а основополагающая: если вдруг завтра проснёмся, а Unix/Linux исчез (вот только вчера был — и нет его . . .), то мало не покажется — в 2008 году только один банк «лопнул» и это привело к мировому экономическому кризису, а если все и всё?

**Специфической особенностью** Linux является открытость исходных текстов ОС — лицензия GPL, по которой распространяется Linux, запрещает закрывать исходники. А это значит, что любой желающий разобраться в том, как устроена эта высокотехнологичная, высоконадёжная ОС, может это сделать. Тем более, что исходники Linux хорошо документированы. Конечно, разобраться в миллионах строк кода — это задача очень нетривиальная. «Порог вхождения» в квалификацию очень высокий. Но возможность-то предоставляется!

По лицензии GPL распространяется не только сама ОС Linux, но и практически всё системное и прикладное ПО дистрибутивов Linux. А это десятки тысяч пакетов программ. Пример: репозитарий ALTLinux (Россия) содержит около 40 тысяч пакетов по состоянию на 2019 год.

А поскольку исходные тексты ОС открыты, то отсюда следует, что эта ОС сама по себе может являться учебным пособием по дисциплинам программирования, причём, учебным пособием очень продвинутым, написанным профессионалами. Аналогичными учебными пособиями являются и десятки тысяч пакетов, распространяемых также по лицензии GPL в составе дистрибутивов. И практически все программы и библиотеки имеют документацию — таково правило включения пакета в дистрибутив. Хотите стать профессиональным программистом? Нет проблем! Открывайте исходники linux-овых программ и самой ОС linux — и учитесь у профессионалов. То есть, действует основополагающий принцип педагогики: «Делай — как я!». Иначе говоря, для целей образования ОС Linux подходит очень хорошо: выполняется основной принцип образовательного процесса: «Делай как я! — делай лучше меня!». Реализуется основополагающее положение обучения по образцу.

Также следует сказать о том, что дистрибутивы Linux распространяются практически бесплатно — по лицензии GPL. То есть, реализуется иная модель бизнеса, нежели при распространении коммерческих дистрибутивов: заработок появляется не при продаже (перепродаже (спекуляции!) — дистрибьюторами) программы, а при сопровождении проданного, а это существенно более высокотехнологичное действие, требующее и гораздо больших знаний и квалификации, и большего количества людей. Сопровождение сложного ПО — это высокие и очень высокие технологии. Сопровождать сложное ПО — это намного сложнее, чем быть просто программистом. Научиться программировать не сложно, программистов миллионы, а разобраться в сложной большой программе — о-о! это нужна квалификация, это не каждый может. Недаром существует у программеров поговорка: «Чем в этой проге разобраться — легче новую написать!».

Следовательно, использование Linux способствует развитию высокотехнологичных отраслей экономики.

И, наконец, очень важный момент: если используется Российский дистрибутив Linux, то и деньги за его разработку, использование и сопровождение остаются в России, то есть, у нас с вами. А это не малые деньги, ведь до сих пор (2019 год) Россия тратит на закупку импортного ПО до $5 (пяти!) млрд. в год.

Поэтому в данном пособии рассматривается в основном операционная система Unix (и прежде всего — Linux), а остальные упоминаются по мере необходимости в целях сравнения. И основным дистрибутивом, рассматриваемом в пособии, является ALTLinux. Почему именно ALTLinux — смотри ниже.

**(Настоятельно!) рекомендуемый авторами** дистрибутив Linux для целей образования (и не только) — это ALTLinux (берётся отсюда: http://ftp.altlinux.ru). Обоснование:

а) это самый хорошо локализованный дистрибутив: не только хорошо локализованы много программ (хорошо локализованый — это переведены и меню программного интерфейса, и программная документация), но и имеется очень много русской документации практически по всем вопросам (например, heap.altlinux.ru или wiki.altlinux.ru); этот пункт, пожалуй, важнейший в обосновании выбора, ибо несмотря на напряжённый труд учителей английского языка в школах и преподавателей кафедр иностранных языков в ВУЗах, знание студентами английского крайне посредственное;

б) самая лучшая поддержка в России, в том числе, бесплатная сообществом ALTLinux на форуме (http://forum.altlinux.org), в рассылках (https://www.altlinux.org/MailingLists), IRC (https://www.altlinux.org/IRC), в социальных сетях (https://telegram.me/alt_linux, https://vk.com/altlinux, https://vk.com/simplylinux, https://www.facebook.com/groups/136328550579/, https://plus.google.com/communities/108911472444655347698) и прочих сервисах (https://www.altlinux.org/Contacts); среднее время ожидания ответа на вопрос — несколько часов; ни один другой дистрибутив linux, претендующий на «русскость», подобной скоростью похвастаться не может;

в) достаточно хорошая распространённость в России, в том числе, в школах;

г) это реально Российский дистрибутив — разработчики живут в РФ и репозитарий находится в России; причём репозитарий ALTLinux — это настоящий отдельный самостоятельный самодостаточный репозитарий с соответствующей инфраструктурой, а не зеркало какого-то там;

д) включен в Реестр (https://reestr.minsvyaz.ru/) — на 2019 год это дистрибутивы: «АлтОбразование-8», «АлтРабочаяСтанция-8», «АлтСервер-8» (https://reestr.minsvyaz.ru/reestr/?sort_by=date&sort=asc&class%5B%5D= 54112&name=alt&owner_status=&owner_name=&set_filter=Y);

е) дистрибутив ALTLinux существует/распространяется как в виде полных сборок (почти всё, что нужно в одном .iso), так и в виде заготовок для создания своих сборок — стартекиты (StarterKit`s), что очень удобно для построения своих оригинальных систем; среди стартеркитов есть сборки и для слабых машин (и даже очень слабых) — https://www.altlinux.org /Starterkits/Memory

ж) ALTLinux, а ныне «Базальт СПО», проводят обучение работе с ОС «Альт» ИТ-специалистов, пользователей, преподавателей и просто жаждущих на различных курсах, семинарах (https://www.basealt.ru/courses /training/), конференциях, вебинарах (https://www.basealt.ru/courses /vebinary/) разного уровня сложности и портале дистанционной поддержки (https://kurs.basealt.ru/). Возможно создание сертифицированных учебных центров на базе вузов. На **портале дистанционной поддержки** содержится много обучающих курсов, в том числе, видеокурсов по ОС «Альт» и прикладным программам дистрибутива, а также методические материалы по их применению в учебном процессе. Обучение, в том числе и бесплатное, с выдачей сертификата. «Базальт СПО» проводит ежегодную конференцию разработчиков свободных программ на базе Калужского ИТ-кластера и ежегодную конференцию «СПО в высшей школе» при поддержке Института программных систем РАН в городе Переславль-Залесский. Сборники тезисов конференций размещаются в **РИНЦ** и на https://books.altlinux.org. Кроме того, «Базальт СПО» является соорганизатором ежегодной научно-практической конференции «OS DAY» (https://osday.ru/);

з) ALTLinux работает на российских процессорах Эльбрус — версии «Альт Рабочая станция» и «Альт Сервер» для отечественной вычислительной платформы «Эльбрус» (**http://mcst.ru/os-alt-na-platforme-elbrus-rossijskaya-os-na-rossijskom-processore**).

**Пособие состоит из нескольких частей**. В теоретической части пособия приведены сведения об организации сетей, о структуре Интернета и именовании в нём, о сетевых сервисах и управлении сетевыми сервисами, а именно:

- лекция 1: краткое содержание материала, читавшегося в дисциплинах бакалавриата «Операционные системы», «Администрирование инфор-

мационных систем», «Управление инфокоммуникационными устройствами связи»;

- в остальных семи лекциях рассматривается устройство Интернета, то есть, стек сетевых протоколов TCP/IP, протокол IP, цифровое именование взаимодействующих объектов в протоколе IP, символическое именование взаимодействующих объектов в системе DNS, сервисы и администрирование сервисов Интернета.

В практической части представлены методические указания к некоторым лабораторным работам по установке, конфигурированию и эксплуатации сетевых сервисов.

Пособие предназначено для практического руководства при проведении преподавателями лекций, семинаров и лабораторных занятий и выполнении заданий студентами указанных специальностей всех форм обучения.

Учебное пособие составлено в соответствии с программой дисциплины «Сетевое программное обеспечение», и предусматривает подготовку магистров, инженеров и бакалавров по направлениям 11.03.02 «Инфокоммуникационные технологии и системы», 09.03.02 «Информационные системы и технологии», 09.03.03 «Прикладная информатика», 02.03.03 «Математическое обеспечение и администрирование информационных систем» и специальности 10.05.01 «Компьютерная безопасность». Может использоваться студентами родственных специальностей и направлений.

# Лекция 1. Общие вопросы организация сетей

В первой лекции кратко повторим тот материал, что вы должны знать из предыдущих дисциплин и который понадобится как основа знаний, в последующих лекциях.

## 1.1. Общие сведения о вычислительных сетях

### 1.1.1. Сети

В экономике (и в жизни вообще) достаточно часто возникает необходимость в передаче информации. Эта задача реализуется с помощью сетей передачи данных.

**Определение.** Сеть передачи данных — совокупность оборудования, а в современных реализациях, ещё и программного обеспечения, предназначенного для передачи информации.

Сети передачи данных бывают телефонные, телеграфные, вычислительные, сети передачи данных между ЦУП и космическими станциями и другие.

**Определение.** Вычислительная сеть — совокупность аппаратно-программного обеспечения предназначенного для передачи данных между компьютерами.

Вычислительные сети получили широкое распространение в последние десятилетия. Более того, они своей функциональностью покрывают все остальные виды сетей, то есть, в современной экономике (и в жизни вообще) они становятся основным видом сетей, а для просто сетей передачи данных невычислительного типа — уже даже сложно привести пример их использования.

## 1.2. Основные определения

### 1.2.1. Классификации

Всё оборудование сетей передачи данных делится на две категории:

- *пассивное* — кабели, разъёмы, патч-панели и др., то есть, то оборудование, которое обеспечивает (поддерживает) передачу сигнала, несущего информацию,

- *активное* — сетевые карты, модемы, коммутаторы, мосты, мультиплексоры и др., то есть, то оборудование, которое генерит или принимает сигнал, используемый для передачи информации.

Кроме того, в сетях передачи данных используется различное вспомогательное оборудование, которое не относится к указанным категориям: это устройства бесперебойного питания, устройства кондиционирования воздуха, монтажные стойки, шкафы, кабель-каналы, средства крепления и другое оборудование, которое не участвует непосредственно в передаче данных.

Активное оборудование требует подачи энергии для генерации/приёма/обработки сигналов: это сетевые карты, повторители, концентраторы, модемы, коммутаторы и др.

Пассивное оборудование подачи энергии не требует: это кабельные системы, соединительные разъемы, коммутационные панели и др.

Обозначения (см. рис. 1):

***DTE** — Data Terminal Equipment, оно же ООД — оконечное оборудование данных.* Таким оборудованием может являться, например, ПЭВМ — ваш персональный компьютер. Иначе говоря, это то оборудование, которое конкретно пользователь использует для обмена данными с другим пользователем: компьютер, мобильник, телеграфный аппарат Бодо, факсовый аппарат и др.

***DCE** — Data Circuit terminating Equipment, оно же АПД — аппаратура передачи данных.* Таким оборудованием может являться, например, сетевая карта в вашем компьютере или модем. Иначе говоря, это то оборудование, которое непосредственно связывает пользовательское оборудование (ООД), например, компьютеры, с линией связи и является, таким образом, пограничным оборудованием.

Обратите внимание, что чёткой границы между ООД и АПД нет. Например, сетевая карта вроде как принадлежит компьютеру, то есть является частью ООД, но с другой стороны она явно является АПД, так как работает непосредственно с кабелем (с линией связи). Аналогично и модем.

***Кабельные системы**, иногда говорят «структурированные кабельные системы»* (СКС, то есть, кабельные системы специальным образом иерархически организованные) — это кабели, патч-панели, коннекторы, розетки, кабель-каналы и др. элементы, с помощью которых строится кабельная система (включая гвозди, шурупы и дюбели! — см. *смету на создание СКС*).

***Промежуточное оборудование*** — это либо оборудование линий связи большой протяжённости (маршрутизаторы, мультиплексоры, повторители и др. оборудование и, конечно, кабельные системы), на которых оно используется как раз для обеспечения этой самой «большой протяжённости», либо это оборудование для организации локальной сети (коммутаторы,

мосты, концентраторы и, опять же, кабельные системы). Промежуточное оборудование это совокупность активного и пассивного оборудования. Работу промежуточного оборудования пользователь не замечает — см. *прозрачность сети*.

***Среда передачи данных*** — промежуточное оборудование, а также просто пространство, если используются беспроводные технологии (радио, инфракрасная или иная оптическая беспроводная техника и др.).

Вырожденный случай — когда для организации обмена данными промежуточное оборудование не используется, например, если в двух ваших компьютерах стоят беспроводные адаптеры BlueTooth и компьютеры стоят на расстоянии не более десяти метров друг от друга.

***Линия связи*** — АПД + среда_передачи_данных + АПД.

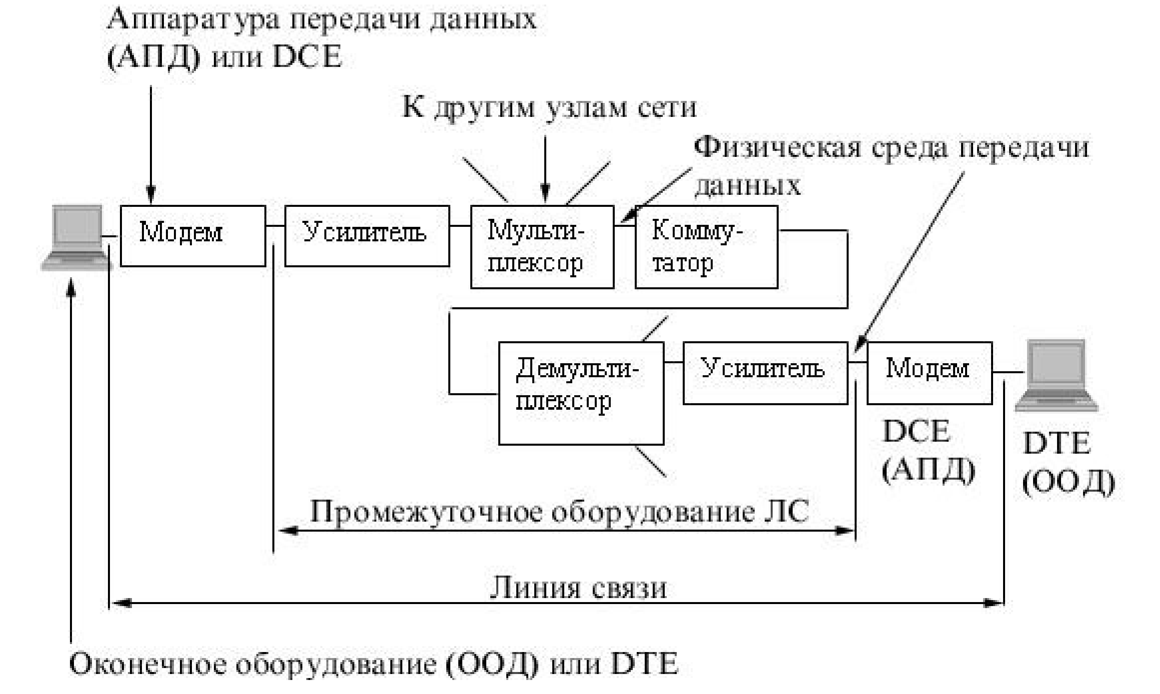



*Рис. 1.* Структура линии связи

Типы оборудования сетей в терминологии ЭМВОС (см. рис. 2):

Конечные системы. ES (End Systems). Являются источниками и/или потребителями информации (ПК, сетевые принтеры, …)

Промежуточные системы. IS (Intermediate Systems). Обеспечивают прохождение информации по сети (концентраторы, маршрутизаторы, модемы, кабельная или беспроводная инфраструктура, соединяющая их).

***Сетевой трафик*** — это поток информации, передаваемый по сети. Трафик кроме полезной информации включает и служебную, необходимую для организации взаимодействия узлов сети.

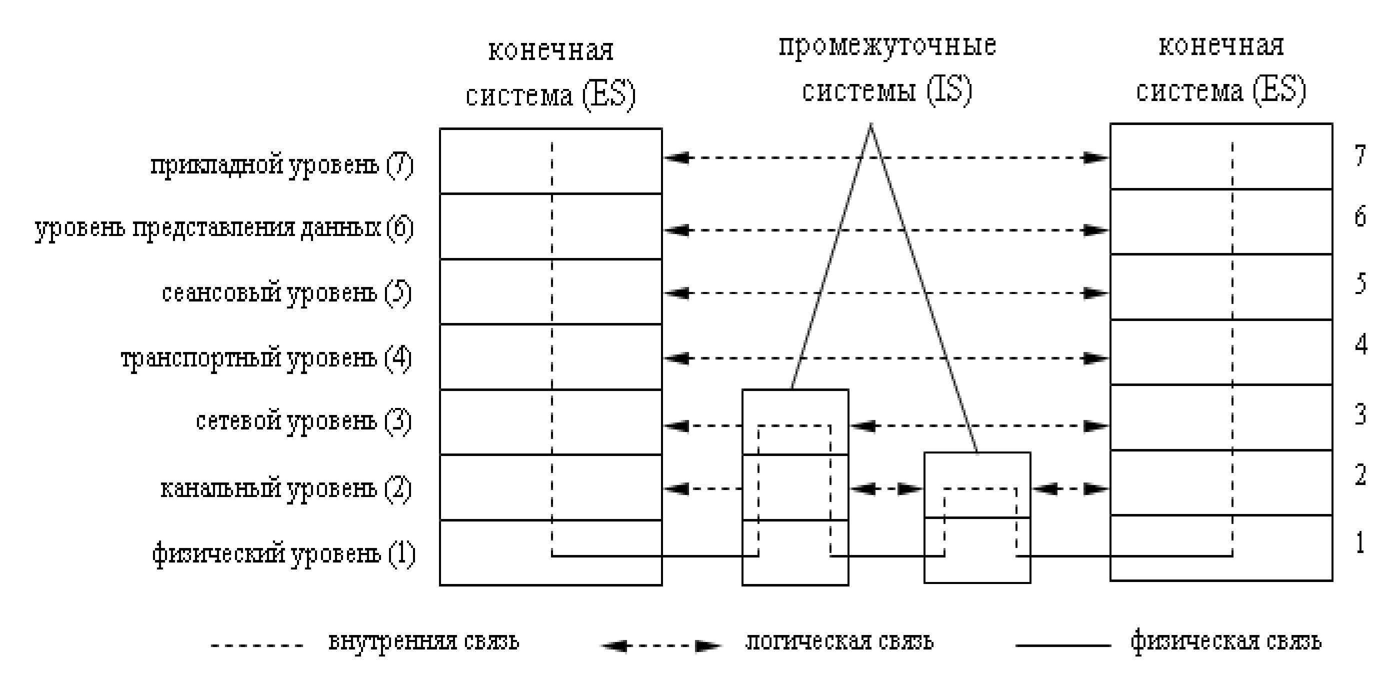



*Рис. 2.* Структура эталонной модели: внутренние связи (по вертикали) реализуются интерфейсами; логические связи (по горизонтали) реализуются протоколами; физические связи (между системами) реализуются через посредство среды (протоколы физического уровня)

***Пропускная способность сети*** определяется количеством информации, проходящей через линию связи за единицу времени (бит/сек, кбит/сек, Мбит/сек, Гб/сек), то есть, трафик в единицу времени по всей линии связи.

***Производительность сети*** применима для активного коммуникационного оборудования. Определяется как общее количество неструктурированной информации, пропускаемой оборудованием за единицу времени (бит/сек), то есть, трафик в единицу времени через активное оборудование.

### 1.2.2. Открытые и закрытые системы

Для организации обмена информацией должен быть разработан комплекс программных и аппаратных средств, распределенных по разным устройствам сети. Поначалу, давным-давно, в 60-70- и даже 80-ые годы каждый разработчик и поставщик сетевых средств сам разрабатывал и реализовывал весь комплекс задач сетевого взаимодействия с помощью собственного набора протоколов, программ и аппаратуры — в экономике есть термин «вертикально организованный рынок». Это приводило к несовместимости этих наборов у разных поставщиков. Такие системы назывались закрытыми. При соединении нескольких закрытых систем используются частные решения, организующие взаимодействие конкретных закрытых систем, которые работают, но ограничивают возможности пользователей,

привязывая их к приложениям или к аппаратному обеспечению определенных производителей.

Открытые системы предполагают реализацию сетевого взаимодействия на основе открытых стандартов, направленных на обеспечение совместимости между различными системами (возможно, разных производителей), то есть, задача построения сетей разбита на несколько взаимосвязанных подзадач с определением правил взаимодействия между ними. Стандартизация этих правил позволяет расширить количество разработчиков аппаратного и программного обеспечения.

### 1.3. Эталонная модель взаимодействия открытых систем (OSI — Open System Interconnection)

ISO — международная организация по стандартизации.

Семиуровневая модель OSI, она же ЭМВОС — Эталонная Модель Взаимодействия Открытых Систем показана на рисунках 2 и 3.

Модель разработана в 1977 г. ISO (International Standard Organization). Она стала основой для определения того, как взаимодействуют системы в сети. В 1984 г. была выпущена новая версия, которая стала международным стандартом (см. рис. 3). Сама модель OSI основана на уровневых протоколах, что позволяет обеспечить:

- логическую декомпозицию сложной сети на обозримые части-уровни;
- стандартные интерфейсы между сетевыми функциями;
- симметрию в отношении функций, реализуемых в каждом узле сети;
- общий язык для взаимопонимания разработчиков различных частей сети.

Функции любого узла сети разбиваются на уровни. Для конечных систем их семь. Внутри каждого узла взаимодействие между уровнями идет по вертикали (по интерфейсам) — см. рис. 2. Взаимодействие между двумя узлами **логически** происходит по горизонтали между соответствующими уровнями (по протоколам). Реально, из-за отсутствия непосредственных горизонтальных связей производится спуск до нижнего уровня в источнике, связь через физическую среду и подъем до соответствующего уровня в приемнике информации. В промежуточных устройствах подъем идет до того уровня, который доступен устройству. Каждый уровень обеспечивает свой набор сервисных функций. Уровень, с которого посылается запрос, и симметричный ему уровень в отвечающей системе формируют свои блоки данных.

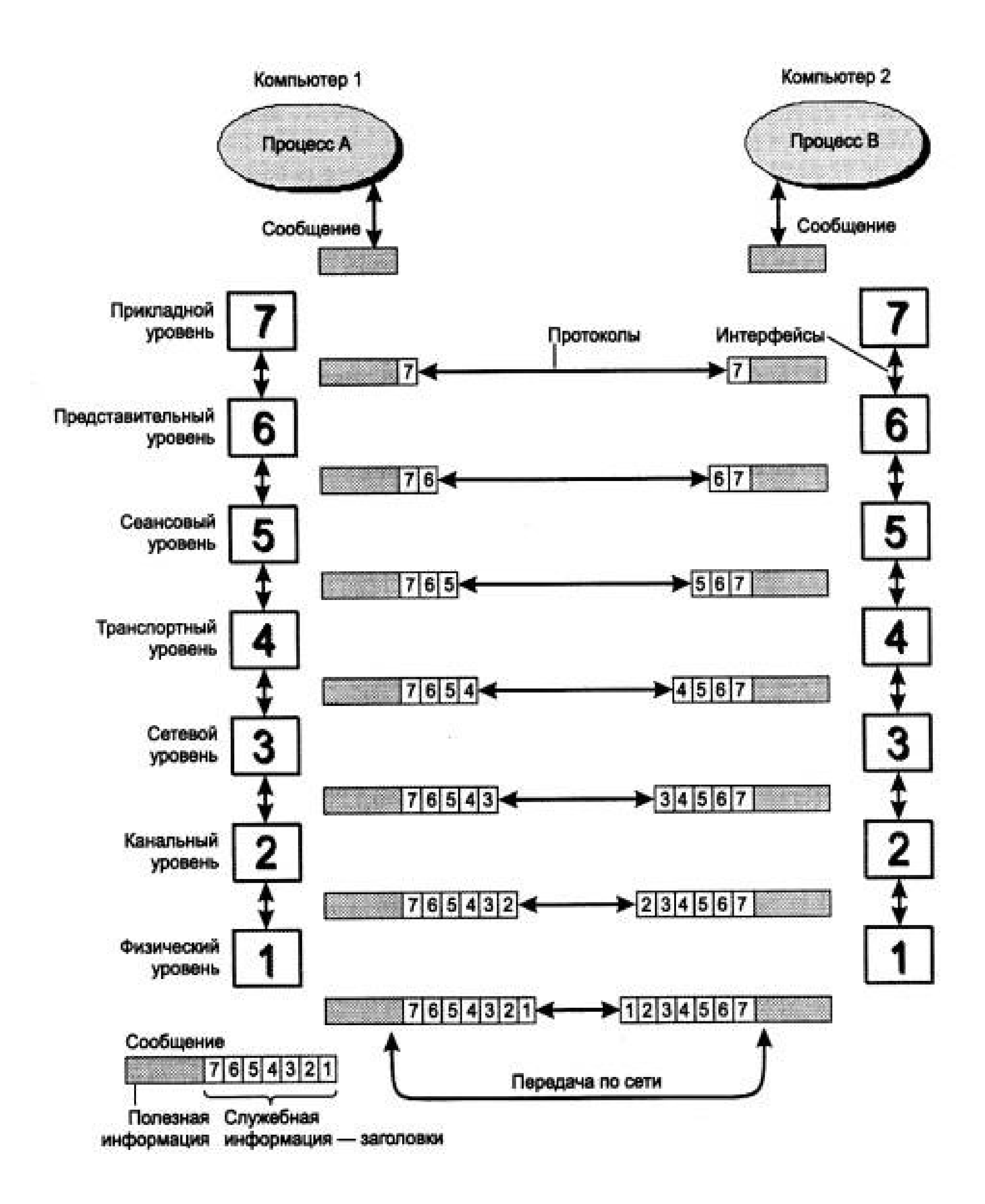



*Рис. 3.* ЭМВОС — Эталонная Модель Взаимодействия Открытых Систем — инкапсуляция пакетов

На каждом уровне определяется свой формат блока данных. Он состоит из заголовка (адрес, управление), поля данных и иногда концевика (например, контрольной суммы по блоку данных). Эти блоки данных по мере спуска вниз по уровням инкапсулируются (вставляются целиком как данные в поле данных — см. рис. 4) в блоки данных нижних уровней. При подъёме происходит обратный процесс — разинкапсуляция.

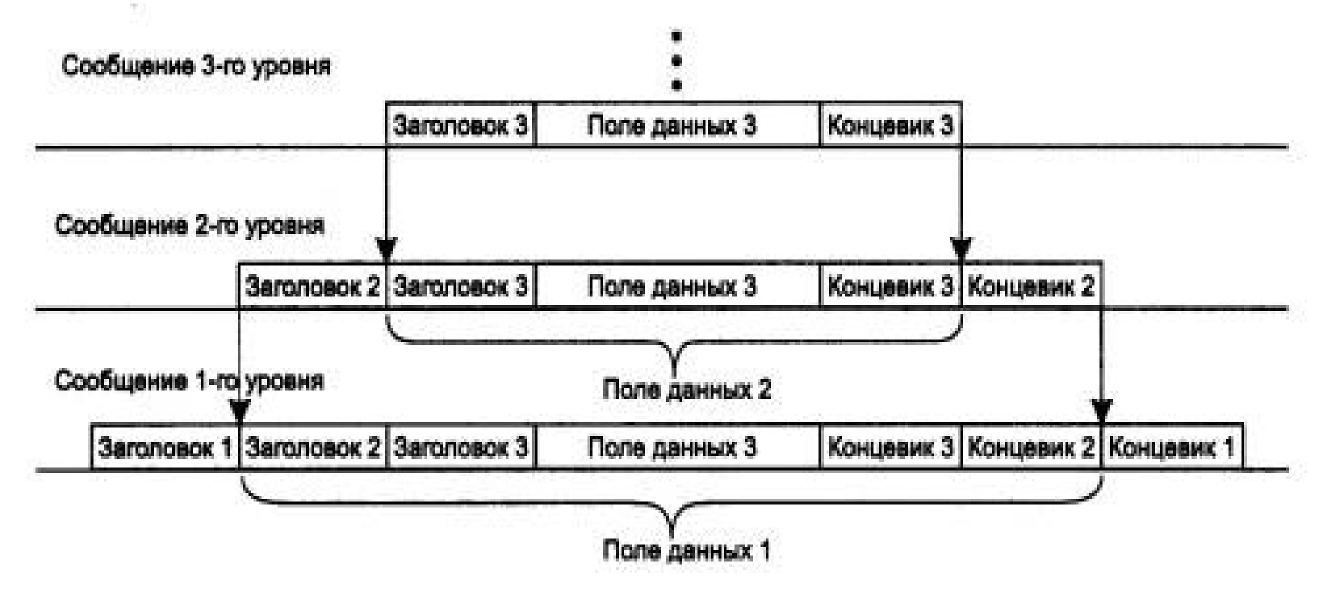



*Рис. 4.* Инкапсуляция

Каждый уровень реализует некоторую функциональность. Она определяется возможностями наборов протоколов и интерфейсов уровня, то есть алгоритмами протоколов и интерфейсов и правилами именования сетевых объектов на этом уровне.

***Протокол*** — формализованные правила, определяющие формат сообщений и алгоритм, по которому обмениваются сообщениями сетевые компоненты, лежащие на одном уровне, но в разных узлах. Формально, протокол — это документ. Но программный модуль, реализующий некоторый протокол, часто для краткости также называют «протоколом». При этом соотношение между протоколом — формально определенной процедурой (документом) и протоколом — программным модулем, реализующим эту процедуру, аналогично соотношению между алгоритмом решения некоторой задачи и программой, решающей эту задачу. Полный протокол должен включать разделы:

- описание формата пакетов данных, формируемых на данном уровне,
- описание алгоритма обмена (работы протокола),
- правила кодирования информации в этом протоколе,

- правила именования сетевых взаимодействующих объектов этого протокола.

Конечно, отнюдь не каждый протокол включает все разделы, большинство протоколов разрабатываются в «контексте» стека протоколов (см. рис. 5).

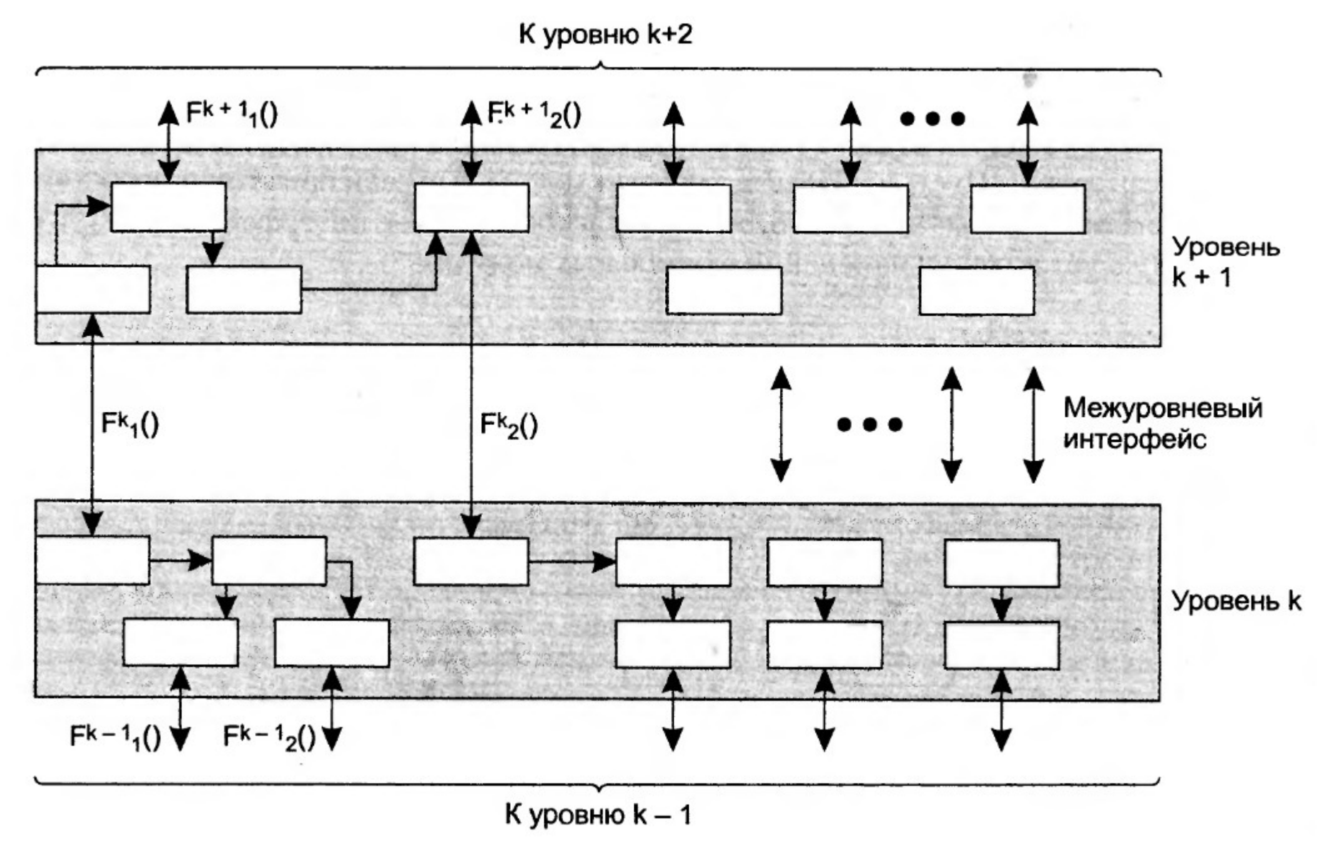



*Рис. 5.* Стек — совокупность взаимосвязанных и иерархически организованных протоколов

***Интерфейс*** — формализованные правила, определяющие формат сообщений и алгоритм, по которому обмениваются сообщениями сетевые компоненты, лежащие в одном узле на соседних уровнях. То есть, интерфейс определяет набор сервисов, предоставляемый некоторым уровнем вышестоящему соседнему уровню. С точки зрения программирования выглядит это как вызов функций некоторой библиотеки. Обратите внимание, что интерфейс всегда предоставляется нижестоящим уровнем вышестоящему уровню. Отсюда такие выражения как «интерфейс человек-ЭВМ» (на самом деле интерфейс предоставляется ЭВМ (нижестоящей «мафыной») вышестоящему человеку). Также взаимодействие сотрудника и начальника определяется интерфейсом сотрудника, который он предоставляет своему начальнику. Этот интерфейс сотрудника определяется «должностной инструкцией», квалификацией и образованием сотрудника.

***Стек коммуникационных (сетевых) протоколов*** — совокупность взаимосвязанных и *и*ерархически организованных протоколов, достаточный для организации взаимодействия узлов в сети. Протоколы стека иерархически организованы — то есть, разбиты на группы-уровни (см. рис. 5), в соответствии со своим предназначением, но, протоколы стека взаимодействуют между собой по вертикали и горизонтали, то есть, они могут обращаться к другим протоколам этого же уровня или к протоколам нижестоящего уровня (по интерфейсу нижестоящего уровня). В некотором смысле каждый протокол стека может рассматриваться как некоторый класс в объектной модели: в протоколе определяются структуры данных (формат пакетов, которыми он обменивается) и алгоритм обмена (что, когда и как должно делаться — методы класса).

## 1.4. Физический уровень

Физический уровень (Physical layer) реализует передачу битов по физическим каналам связи, таким, например, как коаксиальный кабель, витая пара, оптоволоконный кабель или просто пространство. К этому уровню имеют отношение характеристики физических сред передачи данных, такие как полоса пропускания, помехозащищенность, волновое сопротивление и другие. На этом же уровне определяются характеристики электрических сигналов, передающих дискретную информацию, например, крутизна фронтов импульсов, уровни напряжения или тока передаваемого сигнала, тип кодирования, скорость передачи сигналов и др. Кроме этого, здесь стандартизуются типы разъемов и назначение каждого контакта.

Функции физического уровня реализуются во всех устройствах, подключенных к сети. Со стороны компьютера функции физического уровня выполняются сетевым адаптером, последовательным/параллельным портом, контроллером USB и др.

Примером протокола физического уровня может служить спецификация (релиз) 10-Base-T технологии Ethernet, которая определяет в качестве используемого кабеля неэкранированную витую пару категории 3 с волновым сопротивлением 100 Ом, разъем RJ-45, максимальную длину физического сегмента 100 метров, манчестерский код для представления данных в кабеле, а также некоторые другие характеристики среды и электрических сигналов (см. рис. 6).

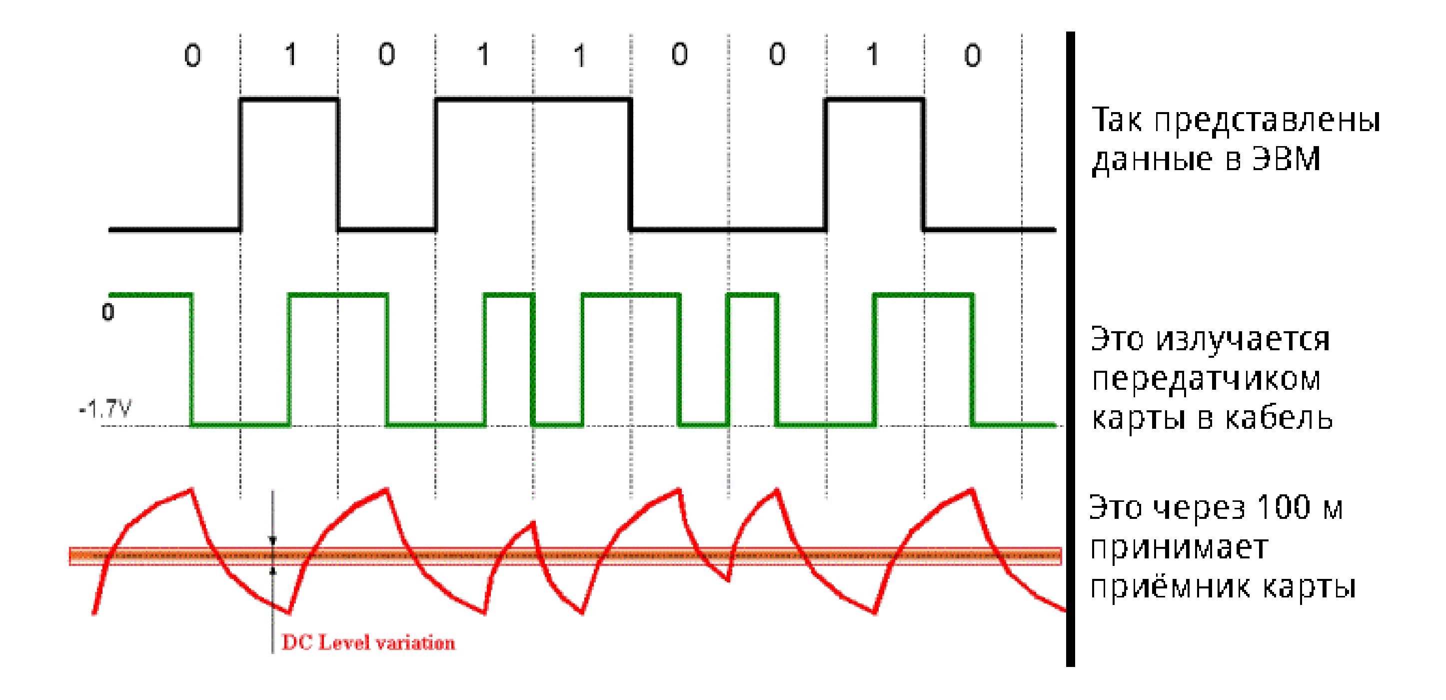



*Рис. 6.* Что излучает сетевая карта отправителя и что получает сетевая карта получателя — сетевая технология Ethernet, спецификация 10Base-T

## 1.5. Канальный уровень: подуровень MAC

### 1.5.1. Подуровень MAC

Канальный уровень делится на два подуровня: нижний подуровень MAC, на котором определяются сетевые технологии, и верхний подуровень LLC, который является единственным и общим для всего (совсем всего) сетевого взаимодействия (рис. 7). Поскольку сетевых технологий много (несколько десятков), то реализаций подуровня MAC тоже много: каждой сетевой технологии соответствует своя реализация подуровня MAC. В данном пункте рассматривается подуровень MAC.

На физическом уровне просто пересылаются биты. При этом не учитывается, что в некоторых сетях, в которых линии связи используются (разделяются) попеременно несколькими парами взаимодействующих компьютеров, физическая среда передачи может быть занята. Поэтому одной из задач канального уровня (Data Link layer) является проверка доступности среды передачи.

Другой задачей канального уровня является реализация механизмов обнаружения и коррекции ошибок. Для этого на канальном уровне биты группируются в наборы, называемые *кадрами (frames* — см. рис. 8*)*. Канальный уровень обеспечивает корректность передачи каждого кадра, по-

мещая специальную последовательность бит в начало и конец каждого кадра, для его выделения, а также вычисляет контрольную сумму, обрабатывая все байты кадра определенным способом и добавляя контрольную сумму к кадру. Когда кадр приходит по сети, получатель снова вычисляет контрольную сумму полученных данных по тому же алгоритму и сравнивает результат с контрольной суммой из кадра. Если они совпадают, кадр считается правильным и принимается. Если же контрольные суммы не совпадают, то фиксируется ошибка.

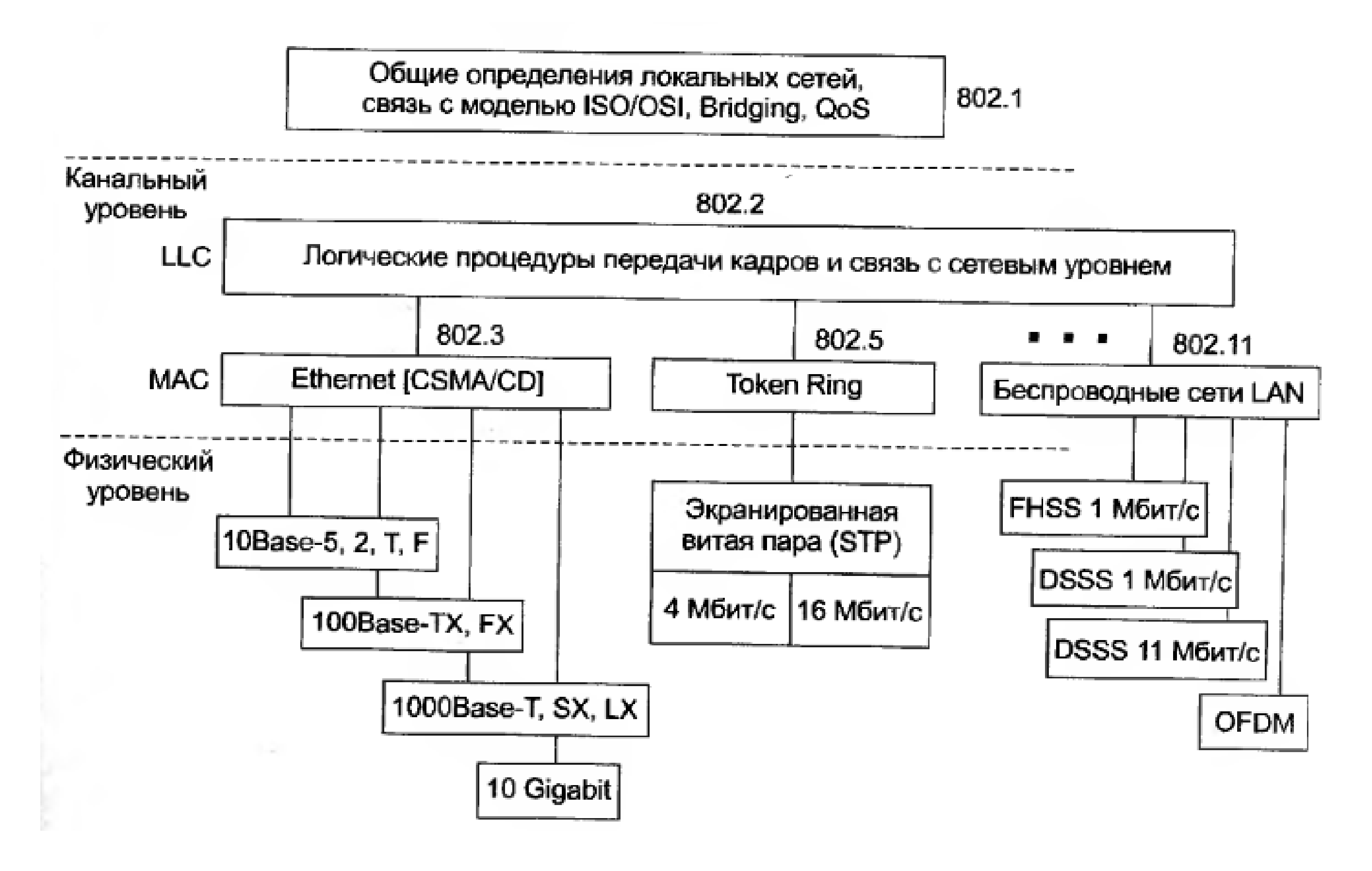



*Рис. 7.* Структура сетевых технологий.
В каждой сетевой технологии показаны далеко не все релизы

Кадр 802.3/LLC

| 6 | 6 | 2 | 1 | 1 | 1(2) | 46–1497 (1496) | 4 |
|---|---|---|---|---|---|---|---|
| DA | SA | L | DSAP | SSAP | Control | Data | FCS |
| | | | Заголовок LLC | | | | |

Рис. 8. Кадр MAC. В середине фигурной скобкой выделен инкапсулированный кадр LLC

Канальный уровень может не только обнаруживать ошибки, но и исправлять их за счет повторной передачи поврежденных кадров. Необходимо отметить, что функция исправления ошибок не является обязательной для канального уровня, поэтому в некоторых протоколах этого уровня она отсутствует, например, в Ethernet и frame relay.

В протоколах канального уровня, используемых в локальных сетях, заложена определенная структура связей (топология) между компьютерами и способы их адресации. Хотя канальный уровень и обеспечивает доставку кадра между любыми двумя узлами локальной сети, он это делает только в сети с совершенно определенной топологией связей, именно той топологией, для которой он был разработан (см. п. 5.5.4.). К таким типовым топологиям, поддерживаемым протоколами канального уровня локальных сетей, относятся общая шина, кольцо и звезда, а также структуры, полученные из них с помощью мостов и коммутаторов. То есть, **алгоритм сетевой технологии определяет топологию сети.** Примерами протоколов канального уровня являются протоколы Ethernet, Token Ring, FDDI, l00VG-AnyLAN и др.

Обращаем внимание на взаимосвязь алгоритма сетевой технологии и топологии сети: **алгоритм первичен**, именно он определяет использование той или иной топологии. Пример: алгоритм Ethernet может работать только на линейных структурах — шина, дерево, звезда в которых ни в коем случае недопустимы кольца; алгоритмы Token Ring, FDDI наоборот, могут работать только на кольцевых структурах; алгоритм l00VG-AnyLAN требует шины с арбитром — специальным устройством, которое определяет порядок (очерёдность) передачи по шине и т. д.

### 1.5.2. Сетевая технология

В середине 90-х годов сформировалось понятие «сетевой технологии» как набора протоколов и интерфейсов физического и канального уровней достаточных для организации взаимодействия узлов в локальной сети. Такими сетевыми технологиями стали Ethernet, Token Ring, FDDI, l00VG-AnyLAN, Frame Relay, ATM и др., в том числе и беспроводные. Эти сетевые технологии иногда называют базовыми сетевыми технологиями.

Каждая сетевая технология определяется некоторым «головным» протоколом канального уровня (Ethernet, Token Ring, FDDI, l00VG-AnyLAN, BlueToosh, MiWi и др. — см. рис. 7), который определяет основные понятия, основной алгоритм технологии. Также в состав сетевой технологии входит набор протоколов и интерфейсов, с помощью которых создаются «релизы» этой сетевой технологии — они также называются спецификациями. Например, сетевую технологию Eternet определяет го-

ловной протокол Ethernet (IEEE 802.3), а «под ним» существуют наборы протоколов, интерфейсов, технических требований и даже стандартов, с помощью которых реализуются различные виды сетевой технологии Ethernet: 10Base-5, 10Base-2, 10Base-T, 10Base-F, 100Base-T, 100Base-TX, 1000Base-T и др., всего несколько десятков различных «релизов» Ethernet.

Также можно сказать, что сетевые технологии — это разные реализации подуровня MAC и физического уровня.

Как правило, сетевые технологии между собой **несовместимы.** Почему? Ну, прежде всего потому, что форматы пакетов (кадров) разные. Во-вторых, и алгоритмы обработки пакетов (кадров) разные. К тому же могут оказаться разным кодирование информации и именование взаимодействующих сетевых объектов. Поэтому сети, построенные на разных сетевых технологиях могут между собой взаимодействовать только через посредство некоторых промежуточных решений, которые переформатируют пакеты (кадры) данных, согласовывают алгортимы, перекодируют информацию и «знают» всех поименно — кто есть кто и в какой сети.

С технологической точки зрения понятием, тесно связанным с понятием сетевой технологии, является понятие «локальной сети».

### 1.5.3. Локальная сеть

Обратите внимание: здесь дано правильное определение локальной сети для ИТ-шников — «технологическое» определение.

***Локальная сеть*** это набор вычислительных систем, сетевое взаимодействие между которыми обеспечивается некоторым релизом некоторой сетевой технологии. Если проще, то это набор компьютеров, объединённых в сеть с помощью некоторой сетевой технологии.

Определение. Локальная сеть — минимальный набор аппаратно-программного обеспечения, достаточный для организации сетевого взаимодействия.

Пример. Предположим, существуют локальные сети по несколько сетевых узлов в каждой: Ethernet (100Base-T), 100VG-AnyLAN и TokenRing. Это разные локальные сети, поскольку в сетевых узлах стоят разные сетевые карты, реализующие разные сетевые технологии, они генерируют кадры данных разного формата, отрабатывают разные алгоритмы и очевидно несовместимы друг с другом.

Сетевые технологии (подуровень MAC), как правило, реализуются аппаратно. Например, в сетевых картах, модемах и других устройствах АПД.

Что нужно для того, чтобы пользователь мог работать в такой локальной сети? Не сильно сложная программа-клиент, которая с верхней

стороны обеспечивала бы интерфейс пользователя (реализовывала команды пользователя по обмену файлами и сообщениями), а с нижней стороны взаимодействовала бы с сетевыми картами. Когда-то (конкретно, в 70-80-90-ые годы, а также в годы всеобщего засилья MS DOS) так оно и было. Можно на одном из сетевых узлов запустить программу, реализующую функциональность ftp-сервера, тогда получим локальную сеть с файловым сервером. Обратите внимание, при этом мы нигде не выходим за пределы понятия «сетевая технология», то есть используются только протоколы и интерфейсы соответствующей сетевой технологии.

**Очень важно!** Таким образом, локальная сеть это самая простая сетевая структура, реализуемая с минимальным набором задействуемых для её организации средств и ресурсов. Все другие сети состоят из локальных сетей, то есть, корпоративные сети, кампусные сети, интернет состоят из локальных сетей, иначе говоря, представляют собой «сети локальных сетей». Такие «сети сетей» называются «глобальными сетями» и для их организации задействуются протоколы более высоких уровней.

### 1.5.4. Топология сети

Топология — способ организации связей между элементами. Различают физическую топологию сети и логическую топологию сети.

***Физическая топология*** — это конфигурация графа, вершинам которого соответствуют компьютеры сети (иногда и другое оборудование, например концентраторы), а ребрам — физические связи между ними (например, кабели).

***Логическая топология*** — это конфигурация графа, вершинам которого соответствуют опять же компьютеры сети (иногда и другое оборудование, например, мосты или коммутаторы), а ребрам — логические связи между ними (например, интерфейсы и протоколы).

Компьютеры, подключенные к сети, часто называют *станциями* или *узлами* сети.

Заметим, что конфигурация *физических связей* определяется электрическими соединениями компьютеров между собой и может отличаться от конфигурации *логических связей* между узлами сети (см. рис. 9). Логические связи представляют собой маршруты передачи данных между узлами сети и образуются путем соответствующей настройки коммуникационного оборудования. Если на рис. 9 концентраторы заменить на коммутаторы, то логическая структура сети будет существенно больше похожа на физическую. На логической топологии мы можем не увидеть некоторых физических устройств в силу свойства «прозрачности сетевого оборудования».

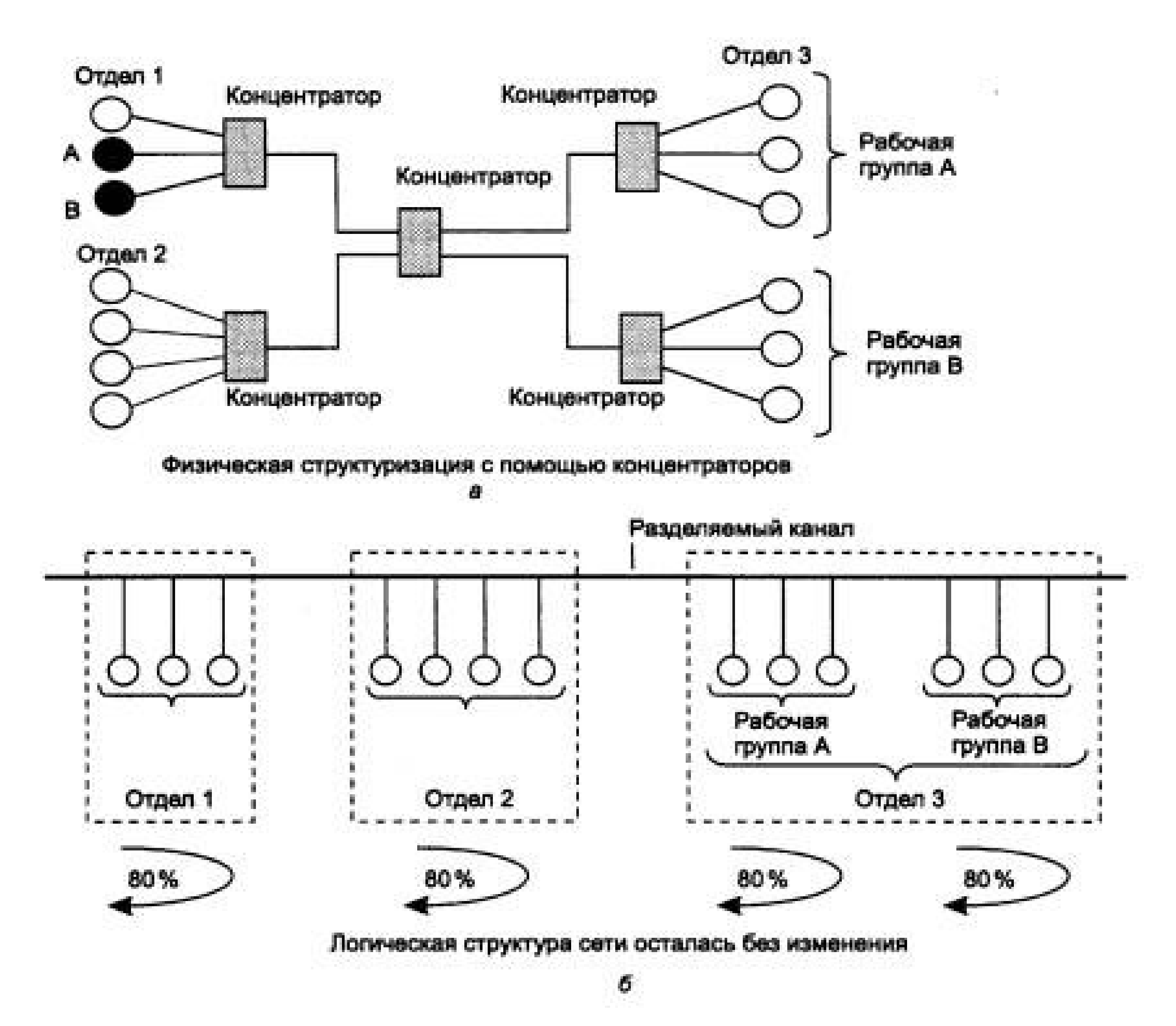



*Рис. 9.* Физическая и логическая структура сети — две большие разницы. 80% трафика в каждом сегменте являются локальными, но концентраторы транслируют трафик на все компьютеры. То есть, в сети с концентраторами каждый компьютер «слышит» любой другой компьютер, подключенный к разделяемому каналу

Типовые топологии (простейшие топологии) — общая шина, звезда, кольцо, точка-точка. Они являются основой локальных сетей и поддерживаются на канальном уровне. Так происходит потому, что основной алгоритм сетевой технологии (алгоритм головного протокола) может работать только в определённых условиях:

- строго определённый способ организации физических связей — топология сети;

- строго определённый способ именования сетевых объектов на этой топологии.

С другой стороны, топология сети определяет особенности алгоритма сетевой технологии: например, может ли линия связи использоваться монопольно или она является разделяемой (shared). Во втором случае возникает комплекс проблем, связанных с её совместным использованием, который включает как чисто электрические проблемы обеспечения нужного качества сигналов при подключении к одному и тому же проводу нескольких приемников и передатчиков, так и логические проблемы разделения во времени доступа к этой линии.

Иначе говоря, алгоритм сетевой технологии (определённый в протоколе) и топология сети определяют друг друга (взаимосвязаны).

Примеры. Алгоритм сетевой технологии Ethernet — CSMA/CD, может работать только на линейных топологиях — шина или точка-точка и топологиях, построенных из них, при этом ни в коем случае не допускаются кольцевые связи (почему?). Алгоритм сетевой технологии TokenRing — алгоритм «маркерного кольца», может работать только на топологии кольцо. Алгоритм сетевой технологии 100VG-AnyLAN — алгоритм «шины с арбитром», требует наличия на шине специального устройства управления доступом к среде — арбитра. И т. д.

В глобальных сетях, которые редко обладают регулярной топологией, канальный уровень часто обеспечивает обмен сообщениями только между двумя соседними компьютерами, соединенными индивидуальной линией связи — топология «точка-точка». Примерами протоколов для этой топологии могут служить широко распространенные протоколы PPP и LAP-B. То есть, линия используется монопольно и основными вопросами являются:

а) надёжность передачи,

б) регулирование трафика.

### 1.5.5. Именование сетевых узлов

С именованием сетевых узлов не возникает проблем только на топологии «точка-точка». Во всех топологиях с разделяемой средой (при подключении к среде передачи трёх и больше компьютеров) возникает проблема их именования — адрес должен быть строго оригинальным (см. рис. 10) в пределах адресуемого (доступного) пространства.

А учитывая, что автономные локальные сети сейчас уже практически не встречаются, то проблема именования становится глобальной. Поэтому, к адресу сетевого узла и схеме его назначения предъявляются определённые требования [27]:

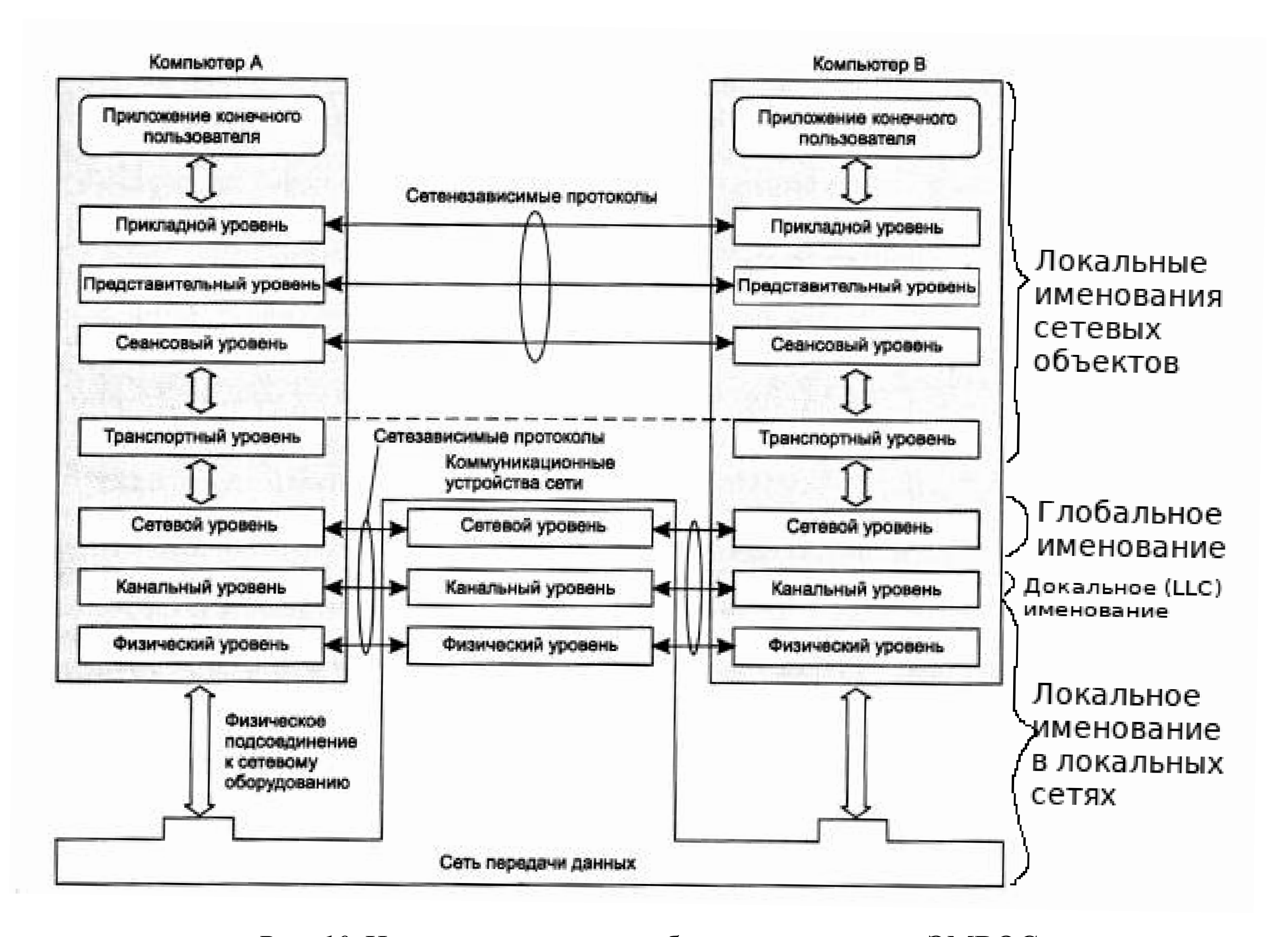



*Рис. 10.* Именование сетевых объектов на уровнях ЭМВОС

1) адрес должен уникально идентифицировать компьютер в сети любого масштаба;

2) схема назначения адресов должна сводить к минимуму ручной труд администратора и вероятность дублирования адресов;

3) адрес должен иметь иерархическую структуру, удобную для построения больших сетей; в больших сетях, состоящих из многих тысяч узлов, отсутствие иерархии адреса может привести к большим издержкам — конечным узлам и коммуникационному оборудованию придется оперировать с таблицами адресов, состоящими из тысяч записей;

4) адрес должен быть удобен для пользователей сети, а это значит, что он должен иметь символьное («человеческое») представление например, www.yandex.ru;

5) адрес должен иметь по возможности компактное представление, чтобы не перегружать память коммуникационной аппаратуры — сетевых адаптеров, маршрутизаторов и т. п.

Исторически сложилось так, что в сетевых технологиях (практически во всех — на подуровне MAC канального уровня) используются «аппарат-

ные» адреса активного оборудования — MAC-адреса, поскольку способ их назначения гарантирует их оригинальность для каждого активного сетевого устройства. Такой способ именования удовлетворяет требованиям 1, 2 и 5 и не удовлетворяет требованиям 3 и 4. Однако в локальных сетях (на уровне сетевых технологий) количество сетевых узлов, как правило, ограничено десятками/сотнями, поскольку есть ограничения в сетевых технологиях на количество сетевых узлов в локальной сети, следовательно такое именование вполне удовлетворительно. А учитывая, что протоколы и интерфейсы сетевых технологий, почти всегда реализуются аппаратно, то в некоторой степени выполняется и требование 4 — не так уж часто человеку-пользователю (не админу) приходится иметь дело с физическими адресами.

Пользователь использует для работы в сети коммуникационное программное обеспечение, которое работает поверх сетевых технологий и, следовательно, есть возможность для организации более «человеческого» именования сетевых узлов и сервисов в сети, что обычно и делается.

## 1.6. Канальный уровень: подуровень LLC

### 1.6.1. Назначение

Протокол LLC обеспечивает для технологий глобальных сетей нужное качество услуг транспортной службы, передавая свои кадры с помощью какой-либо сетевой технологии либо дейтаграммным способом, либо с помощью процедур с установлением соединения и восстановлением кадров.

### 1.6.2. Место протокола LLC в иерархии протоколов

Протокол LLC определяется стандартом ISO 8802.2 (IEEE 802.2) и занимает уровень между сетевыми/транспортными/сеансовыми протоколами (в зависимости от вышестоящего стека сетевых протоколов) и протоколами уровня MAC (сетевыми технологиями).

Вышестоящие протоколы какого-либо стека сетевых протоколов передают через межуровневый интерфейс данные для протокола LLC — свой пакет (например, пакет IP, IPX или NetBEUI), адресную информацию об узле назначения, а также требования к качеству транспортных услуг, которое протокол LLC должен обеспечить. Протокол LLC помещает пакет протокола верхнего уровня в свой кадр, который дополняется необходимыми служебными полями. Далее через межуровневый интерфейс протокол LLC передает свой кадр вместе с адресной информацией об узле назначения соответствующему протоколу уровня MAC, который упаковывает кадр LLC в свой кадр (например, кадр Ethernet — см. рис. 11).

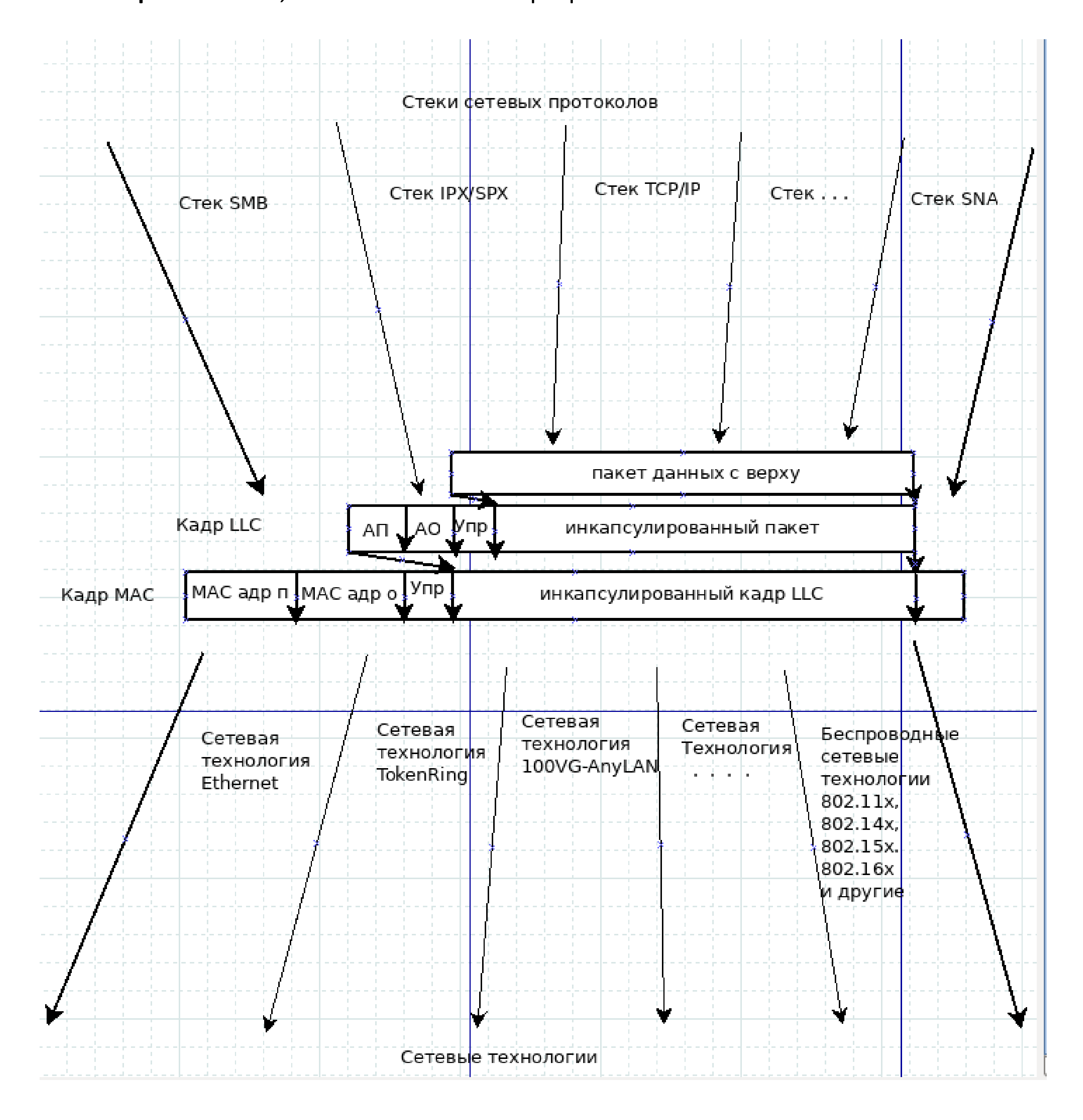



*Рис. 11.* Место протокола LLC в иерархии протоколов. АП — адрес получателя. АО — адрес отправителя. Упр — байты управления. «MAC адр п» — MAC адрес получателя. «MAC адр о» — MAC адрес отправителя

В основу протокола LLC положен протокол HDLC (High-level Data Link Control Procedure), являющийся стандартом ISO. Собственно стандарт HDLC представляет собой обобщение нескольких близких стандартов, характерных для различных технологий: протокола LAP-B сетей X.25 (стан-

дарта, широко распространенного в территориальных сетях), LAP-D, используемого в сетях ISDN, LAP-M, работающего в современных модемах. В спецификации IEEE 802.2 также имеется несколько небольших отличий от стандарта HDLC.

Первоначально в ранних фирменных технологиях подуровень LLC не выделялся в самостоятельный подуровень, да и его функции растворялись в общих функциях протокола канального уровня. Однако, с появлением Интернета и глобальных сетей появилась острая необходимость объединения разношёрстного зоопарка локальных сетей, построенных на основе фирменных сетевых технологий, в сети сетей — корпоративные сети и Интернет. Из-за больших различий в функциях протоколов фирменных технологий, которые можно отнести к уровню LLC, на уровне LLC пришлось ввести три типа процедур. Протокол вышестоящего уровня может обращаться к одной из этих процедур.

### 1.6.3. Три типа процедур подуровня LLC

В соответствии со стандартом 802.2 уровень управления логическим каналом LLC предоставляет верхним уровням три типа процедур:

- LLC1 — процедура без установления соединения и без подтверждения;
- LLC2 — процедура с установлением соединения и подтверждением;
- LLC3 — процедура без установления соединения, но с подтверждением.

Этот набор процедур является общим для всех методов доступа к среде, определенных стандартами 802.3 — 802.5, стандартом на FDDI, стандартом 802.12 на технологию l00VG-AnyLAN, а также AppleTalk, беспроводные сетевые технологии и др.

***LLC1.*** *Процедура без установления соединения и без подтверждения LLC1* дает пользователю средства для передачи данных с минимумом издержек. Это дейтаграммный режим работы. Обычно этот вид процедуры используется, когда такие функции, как восстановление данных после ошибок и упорядочивание данных, выполняются протоколами вышележащих уровней, поэтому нет нужды дублировать их на уровне LLC.

***LLC2.*** *Процедура с установлением соединений и подтверждением LLC2* дает пользователю возможность установить логическое соединение перед началом передачи любого блока данных и, если это требуется, выполнить процедуры восстановления после ошибок и упорядочивание потока этих блоков в рамках установленного соединения. Протокол LLC2 во многом аналогичен протоколам семейства HDLC (LAP-B, LAP-D, LAP-M),

которые применяются в глобальных сетях для обеспечения надежной передачи кадров на зашумленных линиях. Протокол LLC2 работает в режиме скользящего окна.

***LLC3.*** В некоторых случаях (например, при использовании сетей в системах реального времени, управляющих промышленными объектами), когда временные издержки установления логического соединения перед отправкой данных неприемлемы, а подтверждение о корректности приема переданных данных необходимо, используется дополнительная процедура, называемая *процедурой без установления соединения, но с подтверждением LLC3*.

Использование одного из трех режимов работы уровня LLC зависит от стратегии разработчиков конкретного стека протоколов. Например, в стеке TCP/IP уровень LLC всегда работает в режиме LLC1, выполняя простую работу по формированию кадра и мультиплексированию в рамках дейтаграммной передаче и на приёмном конце извлечения из кадра и демультиплексирования пакетов различных протоколов — IP, ARP, RARP и др. (см. рис. 13). Аналогично используется уровень LLC стеком IPX/SPX.

А, вот, стек SMB (Microsoft/IBM), основанный на протоколе NetBIOS/NetBEUI, часто использует режим LLC2. Это происходит тогда, когда сам протокол NetBIOS/NetBEUI должен работать в режиме с восстановлением потерянных и искаженных данных. В этом случае эта работа перепоручается уровню LLC2. Если же протокол NetBIOS/NetBEUI работает в дейтаграммном режиме, то протокол LLC работает в режиме LLC1.

Режим LLC2 используется также стеком протоколов SNA в том случае, когда на нижнем уровне применяется технология Token Ring.

### 1.6.4. Структура кадров LLC

По своему назначению все кадры уровня LLC (называемые в стандарте 802.2 блоками данных — Protocol Data Unit, PDU) подразделяются на три типа — информационные, управляющие и ненумерованные.

***Информационные кадры*** *(Information)* предназначены для передачи информации в процедурах с установлением логического соединения LLC2 и должны обязательно содержать поле информации. В процессе передачи информационных блоков осуществляется их нумерация в режиме скользящего окна.

***Управляющие кадры*** *(Supervisory)* предназначены для передачи команд и ответов в процедурах с установлением логического соединения LLC2, в том числе запросов на повторную передачу искаженных информационных блоков.

***Ненумерованные кадры*** *(Unnumbered)* предназначены для передачи ненумерованных команд и ответов, выполняющих в процедурах без установления логического соединения передачу информации, идентификацию и тестирование LLC-уровня, а в процедурах с установлением логического соединения LLC2 — установление и разъединение логического соединения, а также информирование об ошибках. Все типы кадров уровня LLC имеют единый формат (см. рис. 12).

| | | | | | |
|---|---|---|---|---|---|
| Флаг 01111110 | Адрес точки входа службы назначения (DSAP) | Адрес точки входа службы источника (SSAP) | Управляющее поле (Control) | Данные (Data) | Флаг 01111110 |

*Рис. 12.* Формат кадра LLC

Кадр LLC обрамляется двумя однобайтовыми полями «Флаг», имеющими значение 01111110. Флаги используются на уровне MAC для определения границ кадра LLC. В соответствии с многоуровневой структурой протоколов стандартов IEEE 802, кадр LLC вкладывается в кадр уровня MAC сетевых технологий, то есть, в кадры Ethernet, Token Ring, FDDI и т. д. При этом флаги кадра LLC отбрасываются.

Кадр LLC содержит поле данных и заголовок, который состоит из трех полей:

- адрес точки входа службы назначения (Destination Service Access Point, DSAP);
- адрес точки входа службы источника (Source Service Access Point, SSAP);
- управляющее поле (Control).

*Поле данных кадра* LLC предназначено для передачи по сети пакетов протоколов вышележащих уровней — сетевых протоколов IP, IPX, AppleTalk, DECnet, в редких случаях — прикладных протоколов, когда те вкладывают свои сообщения непосредственно в кадры канального уровня. Поле данных может отсутствовать в управляющих кадрах и некоторых ненумерованных кадрах. Если пакеты данных, пришедшие от вышестоящих уровней небольшие, то протокол LLC может осуществлять мультипликацию (объединение) пакетов в один кадр LLC, а на принимающем узле, соответственно, демультипликацию (выделение) пакетов данных из кадра (? проверить).

*Адресные поля DSAP и SSAP* занимают по 1 байту. Они определяют **именование взаимодействующих объектов** на подуровне LLC и позволяют указать, какой протокол верхнего уровня пересылает данные с помо-

щью этого кадра. Программному обеспечению узлов сети при получении кадров канального уровня необходимо распознать, какой протокол вложил свой пакет в поле данных поступившего кадра, чтобы передать извлеченный из кадра пакет нужному протоколу верхнего уровня для последующей обработки (см. рис. 12).

Для идентификации этих протоколов вводятся так называемые адреса точки входа протокола (Service Access Point, SAP). Значения адресов SAP приписываются протоколам в соответствии со стандартом 802.2. Например, для стека протоколов IBM SNA значение SAP равно 0x4, для протокола IP — 0x6, для протокола NetBIOS (стек SMB) — 0xF0, для стека Banyan — 0xBC, для стека IPX/SPX — 0xE0 и т. д. Для одних протоколов определена только одна точка входа и, соответственно, только один SAP, а для других — несколько. В большинстве случаев адреса DSAP и SSAP совпадают. Например, если в кадре LLC значения DSAP и SSAP содержат код протокола IPX, то обмен кадрами осуществляется между двумя IPX-модулями, выполняющимися в разных узлах. Но в некоторых случаях в кадре LLC указываются различающиеся DSAP и SSAP. Это возможно только в тех случаях, когда протокол имеет несколько адресов SAP, что может быть использовано протоколом узла отправителя в специальных целях, например для уведомления узла получателя о переходе протокола-отправителя в некоторый специфический режим работы. Этим свойством протокола LLC часто пользуется протокол NetBEUI.

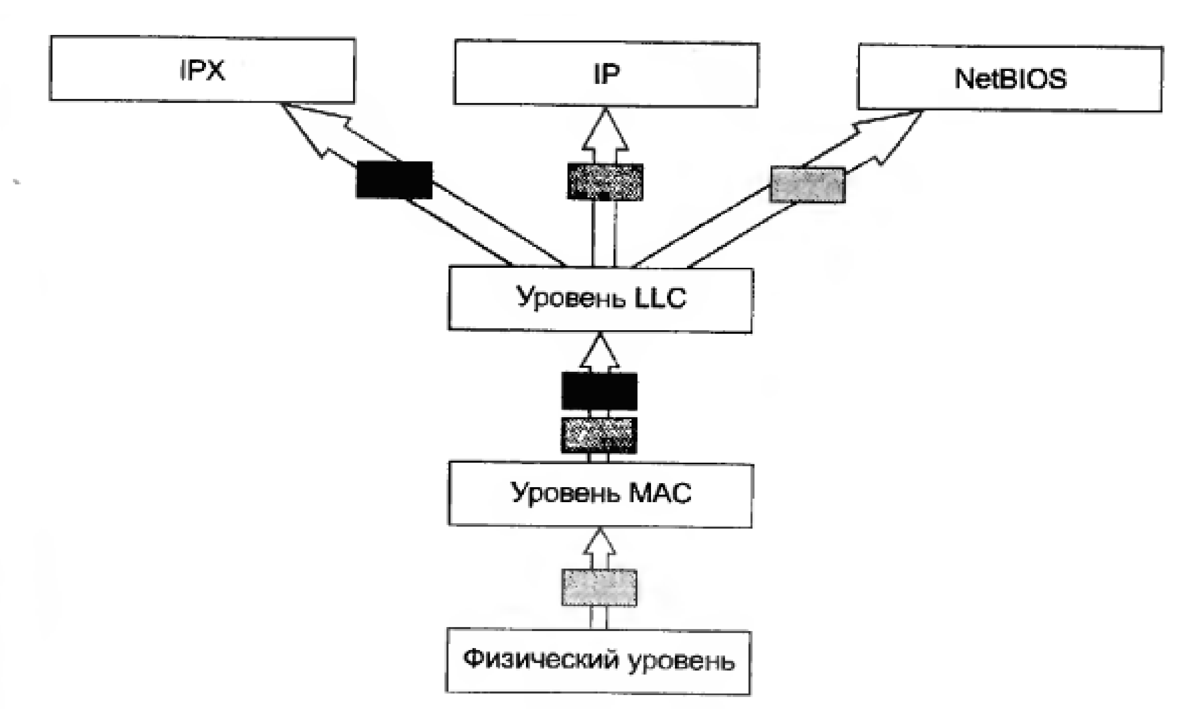



*Рис. 13.* Разинкапсуляция и демультиплексирование кадров протоколом LLC

Таким образом, в качестве имён сетевых узлов на подуровне LLC используются имена вышестоящих протоколов и эти имена определяются тем, какие стеки сетевых протоколов поддерживаются операционной системой, установленной на данном сетевом узле. И, несмотря на то, что имена протоколов (коды протоколов) инвариантны — они определяются централизованно («глобально»), тем не менее, само именование объектов в конкретном сетевом узле (состав протоколов узла) является локальным и зависят от того, какие стеки сетевых протоколов поддерживает ОС данного узла. Пример: на компьютере — Windows, на системной плате — встроенный порт Ethernet, в слот системной платы вставлена карта WiFi- модуля, а в USB-порты — модем GSM и модем BlueTooth. Тогда на компьютере используются как минимум четыре сетевых технологии и два стека сетевых протоколов, то есть, он подключен одновременно к четырём локальных сетям и может быть настроен как маршрутизатор.

*Поле управления* (1 или 2 байта) имеет сложную структуру при работе в режиме LLC2 и достаточно простую структуру при работе в режиме LLC1 (рис. 14).




*Рис. 14.* Структура поля управления кадров LLC

## 1.7. Стеки сетевых протоколов

### 1.7.1. ЭМВОС и структура стеков протоколов

Стеки сетевых протоколов определяются на 3-7 уровнях ЭМВОС. Стеки протоколов появились давно — в 70-80-ые годы XX-го века, поскольку в те далёкие годы вычислительная техника производилась фирмами/организациями в соответствии со своими представлениями о том, что такое сеть и как она должна работать. Ни о какой унификации и совместимости никто не думал — то есть, имел место быть вертикально организованный рынок. В результате появились десятки реализаций межкомпьютерного информационного взаимодействия — сетей, не совместимых друг с другом и на аппаратном, и на программном уровнях. Эти реализации

представляли собой стеки сетевых протоколов («фирменных»), обеспечивавших межкомпьютерное взаимодействие и включавших полностью всю функциональность от физического до прикладного уровней.

Однако в 90-ые годы с переходом от локальных к корпоративным, а затем и к глобальным сетям, появилась острая необходимость в обеспечении совместимости аппаратного и программного обеспечения сетей. Реализовывалась совместимость посредством унификации и стандартизации интерфейсов и протоколов и при этом важную роль играла также цена реализаций и цена обеспечения совместимости.

В итоге к концу 90-ых годов оформилась новая структура сетевого взаимодействия. В основу структуры был положен стек сетевых протоколов OSI, разработанный в 80-ые годы для реализации на мейнфреймах. Поскольку разрабатывался он для больших ЭВМ, то никаких особых ограничений при его разработке не выдвигалось и он получился сложным, но полнофункциональным и хорошо документированным. Но 90-ые годы — это годы бума ПЭВМ, что привело к тому, что мейнфреймы стали экзотикой, а для слабых ПЭВМ 90-х годов стек OSI был слишком сложным для реализации. В результате стек OSI в силу свой полнофункциональности и документированности был определён как Эталонная Модель Взаимодействия Открытых Систем — ЭМВОС.

В соответствии с этой моделью вся структура сетевого взаимодействия делилась на две группы уровней: нижнюю группу образовывали 1-ый и 2-ой уровни Эталонной Модели и в этой группе определялось взаимодействие в локальных сетях, то есть, разнообразные аппаратные реализации, определявшие физическую организацию сетей. Эти нижние 1 и 2 уровни ЭМВОС в 90-ые годы оформились и унифицировались в рамках понятия «сетевые технологии». Они являются базой для понятия «локальная сеть» как минимально необходимой совокупности аппаратно-программного обеспечения для организации сетевого взаимодействия. Алгоритмы на этих уровнях достаточно просты и представимы в виде автоматов, следовательно, могут быть реализованы аппаратно, что и делается в виде различных сетевых устройств: сетевых карт, модемов, мультиплексоров, коммутаторов, концентраторов и т. п. В силу аппаратной реализации эти уровни являются операционнонезависимыми, то есть, для работы с устройствами в ОС необходимы только драйверы, реализующие интерфейс с ними, и более никак эти устройства на работу ОС не влияют.

А, вот, с верхней группой уровней (3-7) ситуация совсем иная. Во-первых, они алгоритмически намного сложнее и в настоящее время не могут быть представлены в виде автоматов, а потому не реализуемы аппарат-

но. А поскольку они реализуются только программно, то это определяет их зависимость от операционной системы, под которую они написаны. Более того, уровни 3, 4 и 5 для обеспечения эффективности и скорости работы реализуются как модули операционных систем и встроены в ядра ОС и только прикладные протоколы реализуются вне ОС, но как программы под определённую ОС. То есть, стек сетевых протоколов всегда операционно-зависим: стек SMB — это Windows, стек IPX/SPX — это NetWare, стек AppleTalk — это Mac OS, стек TCP/IP — это unix, стек SNA — это стек мейнфреймов (OS390 и других) и серверов IBM и т. д. Под некоторые операционные системы (например, unix) существовало несколько стеков сетевых протоколов.

В результате в начале 90-х годов стеков сетевых протоколов существовало столько же, сколько операционных систем и даже больше. А потом пришёл Интернет и всех «выгнал», поскольку Интернет базируется на unix-овом стеке TCP/IP, и потому к настоящему времени из всего многообразия стеков остались только:

- стек TCP/IP — unix/linux и основа Интернета;

- стек SMB — в силу распространённости Windows; MS и рада бы уже от него избавиться, но, ведь, он, «зараза», встроен в ядро ОС, его просто так не отключишь;

- стек AppleTalk — как тяжёлая наследственность на Mac'ах: после перевода ядра ОС на FreeBSD (Mac OS X — уже TCP/IP) фирме Apple приходится поддерживать стек AppleTalk как наследуемый;

- стек SNA — можно встретить на мейнфреймах IBM — если вы с ними встретитесь.

Причём, поскольку Интернет (и корпоративные сети) базируется на unix-овом стеке TCP/IP, то для обеспечения работы с Интернетом (и корпоративными сетями) стек TCP/IP должен быть реализован под другие ОС, если таковые используются на компьютерах по каким-либо причинам. И это действительно так: стек TCP/IP реализован как подключаемый модуль для других ОС.

Все остальные стеки сетевых протоколов «вымерли». Причём, самая интересная ситуация была со стеком IPX/SPX фирмы Novell: в середине 90-х годов почти 70 % всех локальных и корпоративных сетей работали на NetWare, но большая часть — нелегально, поскольку для легального использования NetWare необходимо было регистрироваться в Novell и покупать у неё номера сетей. К сожалению, руководство Novell не осознало, чем оно владеет, и не сделало регистрацию и выделение номеров сетей бесплатным процессом — и где теперь Novell? Следствие: Интернет бази-

руется на TCP/IP, хотя всё было готово к тому, чтобы он базировался на стеке IPX/SPX.

### 1.7.2. Отличительные особенности стеков

Как уже было сказано, стеки сетевых протоколов являются операционнозависимыми. Это прежде всего означает, что они разрабатывались и писались в рамках разработки соответствующих операционных систем и в соответствии с теми взглядами и представлениями на сетевое взаимодействие, которое имели разработчики ОС. Как следствие, они получились разными и отличаются друг от друга также, как и сами операционные системы, и также несовместимы (см. рис. 15).

<table>
<tr><th>ЭМВОС (Модель OSI)</th><th>Стек SMB IBM/ Microsoft (Windows, OS/2)</th><th>Стек TCP/IP (unix, linux)</th><th>Стек IPX/SPX Novell (NetWare)</th><th>Стек OSI</th><th>Стек AppleTalk (Mac OS)</th><th>Стек SNA (OS-390 и др. ОС больших ЭВМ)</th></tr>
<tr><td>7. Прикладной</td><td rowspan="2">SMB и др.</td><td rowspan="2">Telnet, FTP, SNMP, SMTP, POP, IMAP, HTTP, Torrent и др.</td><td rowspan="2">NCP, SAP и др.</td><td>X400, X.500, FTAM</td><td>AFP, и др.</td><td>DIA, SNADS, DDM, и др.</td></tr>
<tr><td>6. Представительный</td><td>Представительный протокол OSI</td><td></td><td>IMS, ISO CICS, и др.</td></tr>
<tr><td>5. Сеансовый</td><td rowspan="2">NetBIOS/NetBEUI и др.</td><td></td><td></td><td>Сеансовый протокол OSI</td><td>ZIP, ADSP, ASP, PAP, и др.</td><td>AAPC, VTAM, и др.</td></tr>
<tr><td>4. Транспортный</td><td>TCP, UDP и др.</td><td>SPX и др.</td><td>Транспортный протокол OSI</td><td>RTMP, NBP, AEP, AURP, ATP, и др.</td><td>APPN, и др.</td></tr>
<tr><td>3. Сетевой</td><td>нет</td><td>IP, RIP. OSPF, ICMP, IGMP, ARP, и др.</td><td>IPX, RIP, NLSP и др.</td><td>ES-ES, IS-IS</td><td>DDP, AARP, и др.</td><td>NCP, и др.</td></tr>
<tr><td rowspan="2">2. Канальный</td><td colspan="6">2.1. “Окно”: 802.2 LLC (LLC1, LLC2, LLC3)</td></tr>
<tr><td colspan="6">2.2. Сетевые технологии: 802.3x (Ethernet), 802.5 (Token Ring), FDDI, SLIP, 802.12 (100VG-AnyLAN), X.25, ATM, LAP-B, LAP-D, PPP, 802.11x, 802.14x, 802.15x, 802.16x и другие</td></tr>
<tr><td>1. Физический</td><td colspan="6">Среда: Коаксиал, экранированная и неэкранированная витая пара, оптоволокно, радиоволны и др.</td></tr>
</table>

*Рис. 15.* Структура стеков протоколов

То есть, первыми отличиями стеков друг от друга являются:

- состав протоколов стеков;
- их функциональное назначение.

Кроме того, очень большое значение имеют принципы именования сетевых объектов, реализованные в стеках протоколов и прежде всего на третьем (сетевом) уровне. Эти принципы определяют не только алгоритмы работы протоколов сетевого уровня, но и возможности логической организации сетей в том или ином стеке, то есть, то, как выглядят (как строятся) крупные сетевые структуры. Очевидно, что в реальной экономике в корпоративной среде, а также в иных (неэкономических) крупных организациях возможности по построению крупных сетевых структур играют большую роль и определяют применимость стеков протоколов.

### 1.7.3. Именование узлов сети в стеке SMB

Физический адрес узла = MAC-адрес, например: 00:04:79:66:0A:46. Почти всегда он представляется в 16-ричном виде. Первые три байта идентифицируют фирму-производителя данного сетевого устройства, а последние три байта — порядковый номер данного вида устройств, выпущенных этой фирмой. Так получается, что каждое активное устройство сети имеет свой индивидуальный номер и не существует устройств, имеющих одинаковый MAC-адрес.

Этому физическому адресу в стеке SMB присваивается алфавитно-цифровое (символьное) имя, которое определяется на этапе установки Windows.

Физический адрес (MAC-адрес) предназначен для использования аппаратным и программным обеспечением (когда они формируют пакеты для отправки или обрабатывают принятые пакеты).

Символьное имя этого адреса предназначено для использования человеком.

В стеке SMB символьное имя «привязывается» к MAC-адресу, является его синонимом (алиасом). Это видно здесь:

- «Пуск → Панель управления → Сетевые подключения → Подключение по локальной сети»; в новом окне «Подключение по локальной сети — Состояние» на вкладке «Поддержка» выбрать «Подробности» — увидим привязку IP-адреса к MAC-адресу на данном копьютере;

- а в новом окне «Подключение по локальной сети — Свойства» на вкладке «Общие» выбрать «Протокол Интернета (TCP/IP)», нажать клавишу «Свойства» — увидим назначение свойств стека TCP/IP на данном компьютере.

А, вот, где имя компьютера (символьное) привязано к MAC-адресу?

### 1.7.4. Именование узлов сети в стеке IPX/SPX

Адрес узла сети (или имя узла — в этом стеке адрес и имя не различаются, это одно и то же) в стеке IPX/SPX двухуровневое: на нижнем уровне используется физический адрес узла (MAC-адрес размером 6 байт, например: 00:50:22:B5:7E:93), на верхнем уровне (через точку) — номер сети размером в 4 байта. То есть, в качестве номера сети служит некоторое 16-ричное число, например, A311. Формально, оно должно выдаваться фирмой Novell по запросу системного администратора, а обычно оно выбирается системным администратором по некоторым ему известным критериям.

Физический адрес узла (MAC-адрес) в стеке IPX/SPX служит также именем компьютера.

Надстройкой над этой двухуровневой структурой имён служит NDS (NetWare Directory Service). В сервисе NDS свои правила именования (иерархические), но они распространяются не только на узлы сети, но и на другие объекты распределённой вычислительной системы (пользователей, элементы организационной структуры фирмы/учреждения, сервисы в системе и т. д.). Именование объектов в NDS ориентировано на человека и, как правило, отражает организационную структуру того социально-экономического объекта, в котором функционирует сеть NetWare. То есть, сервис NDS хотя и привязан к вычислительной сети (узлы сети в нём тоже отражены), но в целом является существенно более абстрактной системой, нежели вычислительная сеть, которая в этом сервисе оказывается очень глубоко скрытой социально-экономическими наслоениями.

Таким образом, в стеке протоколов IPX/SPX числовые адреса узлов сети (например, **00:50:22:B5:7E:93.A311**) используются программами и оборудованием (и иногда системными администраторами — когда они знают об их существовании), а символьные имена NDS предназначены для использования человеком.

### 1.7.5. Именование узлов сети в стеке TCP/IP

В этом стеке протоколов используется двухуровневая система именования узлов сети, которая выглядит как трёхуровневая.

Физическим адресом узла является 6-байтовый MAC-адрес. Этому физическому адресу присваивается имя узла, которое состоит из двух составляющих — символьной (например, comp1.lab213.ulsu.ru) и числовой (например, 192.168.99.11). То есть, в этом стеке имя сетевого узла имеет две формы: символьную и цифровую.

Символьная составляющая, длиной до 255 байт, имеет иерархическую (многоуровневую) структуру и определяется доменной структурой Internet'a. Она состоит из имени хоста (hostname, например, comp1) и доменной части имени (например, lab213.ulsu.ru). Имя хоста присваивается, как правило, администратором хоста (например, при установке ОС или при настройке сети). Доменная часть имени определяется вышестоящими администраторами соответствующих доменов и (иногда) регистрируется в InterNIC, международной организации, ведающей доменной структурой Internet'a.

Цифровая часть имени, размером 4 байта, имеет двухуровневую структуру — <IP-адрес сети><IP-адрес хоста> и назначается администратором по некоторым правилам назначения IP-адресов и также (иногда) может быть зарегистрировано в IANA

В целом назначение имени узла в стеке протоколов TCP/IP определяется спецификацией DNS (Domain Name System), определённой стандартами RFC 1034 и 1035. DNS — распределённая база данных, поддерживающая иерархическую систему имён для идентификации узлов в сети для автоматического поиска IP-адреса по известному символьному имени узла сети Internet и обратно, символьного имени узла по известному IP-адресу.

Соответствие между физическим адресом узла сети и именем узла сети (точнее, числовой частью имени узла сети — IP-адресом), обеспечивается специальным сервисом локальной сети — ARP/RARP, работающим на границе канального и сетевого уровней, а также работающими поверх них сервисами BOOTP и DHCP. В каждом узле сети ведётся ARP-таблица соответствий известных данному узлу MAC-адресов соседних узлов и их IP-адресов.

Таким образом, в стеке протоколов TCP/IP числовые адреса узлов сети (MAC-адреса и IP-адреса) используются программами и оборудованием (и иногда системными администраторами — когда они знают об их существовании), а символьные имена предназначены для использования человеком — администраторами и обычными пользователями. Случаи использования пользователями IP-адресов для доступа к узлам сети свидетельствуют о неисполнении системными администраторами своих обязанностей на почве некомпетентности.

### 1.7.6. Следствия из именования

**Следствие 1:** Определения локальной сети в стеках протоколов.

1. Стек SMB: Локальная сеть — это

а) все узлы, что доступны физически (по MAC-адресам).

*Примечание 1.* Для того, чтобы увидеть «чистую» сеть Windows, обеспечиваемую этим стеком протоколов, необходимо в сетевых настройках компьютеров удалить «протокол TCP/IP»; тогда в «сетевом окружении» увидим «чистую» локальную сеть Windows. Именно локальную сеть — никаких Интернетов, конечно, не будет.

*Примечание 2.* Таким образом, локальная сеть в понимании стека SMB — это совокупность компьютеров, подключенных к одной кабельной системе, на которой используется некоторая сетевая технология: Ethernet, TokingRing, 100VGanyLAN, FDDI и т. д. Например, если кабельная система — кольцо и на нём реализуются сетевые технологии TokenRing или FDDI, то локальная сеть это совокупность компов, подключенных к этому кольцу и только этих компов. Если кабельная система шина, звезда или дерево, то есть, кабельная система, удовлетворяющая требованиям топологии Ethernet, то локальная сеть в понимании стека SMB — это совокупность компов, подключенных к этой кабельной системе и только этих компов. И хотя использование коммутаторов позволяет снять или, по крайней мере, снизить жёсткость некоторых ограничений, всё равно построить большую кабельную систему с сетевой технологией Ethernet невозможно — упираемся в ограничение технологии Ethernet — не более 1024 компов в сети и кабельную систему приходится разбивать на сегменты.

2*. Стек IPX/SPX: Локальная сеть — это

а) те узлы, что доступны физически (по MAC-адресам),

б) и только те из них, которые имеют одинаковый номер сети N.

*Примечание 1.* Таким образом, в стеке IPX/SPX решающими являются не только ограничения кабельной системы (см. выше стек SMB), но и логическое ограничение — принадлежность компа к логической совокупности — «номер сети N», то есть, ограничение, связанное с принципами именования сетевых узлов в данном стеке сетевых протоколов.

3**. Стек TCP/IP: Локальная сеть — это

а) те узлы, что доступны физически (по MAC-адресам),

б) и только те из них, которые имеют одинаковую доменную часть имени (входят в один и тот же поддомен),

в) и только те из них, которые имеют ip-адреса из одной сетки адресов.

*Примечание 1.* Таким образом, в стеке TCP/IP решающими являются не только ограничения кабельной системы (см. выше стек SMB), но и два логических ограничения — принадлежность компа к логическим совокуп-

ностям — «поддомену», который определяется правилами символьного именования сетевых узлов в данном стеке сетевых протоколов и сетке ip-адресов, которая определяется правилами цифрового именования сетевых узлов в данном стеке сетевых протоколов.

**Следствие 2:** Понятие корпоративной сети в стеках протоколов.

*Определение.* Копоративная сеть — объединение локальных сетей в более сложную конструкцию «сеть сетей». Этот процесс (объединение) реализуется с помощью использования специальных сетевых устройств — маршрутизаторов, работающих на третьем (сетевом) уровне ЭМВОС.

1. Стек SMB: Корпоративная сеть — это
- то же, что локальная.
То есть, понятие «корпоративная сеть» формально отсутствует.

2. Стек IPX/SPX: Корпоративная сеть — это
- объединение («прямая сумма») локальных сетей:
N (+) . . . (+) router's
где N — локальная сеть по пункту 2*,
router's — сетевые устройства — маршрутизаторы, обеспечивающие объединение локальных сетей в более сложную конструкцию «сеть сетей», то есть корпоративную сеть.

3. Стек TCP/IP: Корпоративная сеть — это
- объединение («прямая сумма») локальных сетей:
M (+) . . . (+) router's
где M — локальная сеть по пункту 3**,
router's — сетевые устройства — маршрутизаторы, обеспечивающие объединение локальных сетей в более сложную конструкцию «сеть сетей», то есть корпоративную сеть.

*Примечание 1.* В силу того, что в стеке SMB понятие «корпоративная сеть» отсутствует, но оно необходимо для пользователей, MS вынуждено надстроить над локальной сетью конструкцию «домен+Актив_Директори» для того, чтобы искусственно создать необходимую функциональность.

*Примечание 2.* Однако, интересно то, что этот надстроенный функционал в своей работе опирается на возможности стека протоколов TCP/IP, включаемого на Windows, и без него самостоятельно работать не может (!).

**Вывод**, очень интересный и важный: А мир-то реально держится на unix'ах! (!) Windows без unix самостоятельно шагу ступить не может!

И не забудьте, что первая вычислительная сеть была создана в СССР ещё в середине 60-х в рамках проекта по созданию системы ПРО. То есть, на несколько лет раньше, чем аналогичное было создано в Штатах.

**Вопросы «на засыпку»**

1. Канал с какой пропускной способностью арендовал клиент у провайдера, если Total Commander показал скорость копирования файлов 2.3-2.5 мегабайта в секунду?

2. Почему недопустимы кольцевые структуры (петли) в кабельной системе, если кабельная система используется сетевой технологией Ethernet?

3. Почему ограничивается длина кабеля в кабельных системах?

4. Какая разница между хабом и свичём?

5. Что такое Ethernet?

6. Для чего нужна контрольная сумма в кадрах Ethernet?

7. Ваш смартфон является ООД или АПД?

8*. Предположим, в конторе вы создаёте сеть. Вам предоставили Гигабитный коммутатор, кабель 6-ой категории и прочие комплектующие. Но вдруг выяснилось, что один из абонентов находится на расстоянии 150 метров от коммутатора и невозможно ни коммутатор к нему придвинуть, ни абонента к коммутатору. На покупку дополнительного свича/хаба денег нет. Как всё-таки подключить этого абонента — предложите самое дешёвое решение.

# Лекция 2. Стек TCP/IP

## 2.1. Структура стека протоколов TCP/IP

### 2.1.1. Уровни стека

В стеке TCP/IP определены 4 уровня (см. рис. 16), каждый из которых несет на себе некоторую нагрузку по решению основной задачи — организации надежной и производительной работы составной сети, части которой построены на основе разных сетевых технологий.

*Замечание 1.* Обратите внимание на формулировку: «на основе разных сетевых технологий».

| | |
|---|---|
| Уровень I | Прикладной уровень |
| Уровень II | Основной (транспортный) уровень |
| Уровень III | Уровень межсетевого взаимодействия |
| Уровень IV | Уровень сетевых интерфейсов |

*Рис. 16.* Многоуровневая архитектура стека TCP/IP

Основой всей архитектуры стека является *уровень межсетевого взаимодействия*, который реализует концепцию передачи пакетов в режиме без установления соединений, то есть, дейтаграммным способом. Именно этот уровень обеспечивает возможность перемещения пакетов по сети, используя тот маршрут, который в данный момент является наиболее рациональным. Этот уровень также называют уровнем internet, указывая тем самым на основную его функцию — передачу данных через составную сеть.

*Замечание 2.* Обратите внимание на формулировку: «передачу данных через составную сеть».

Замечания 1 и 2 имеют прямое отношение к цели создания и назначению стека TCP/IP и эффективности его использования в реальных условиях.

### 2.1.2. Уровень межсетевого взаимодействия

Основным протоколом сетевого уровня (в терминах модели OSI) в стеке является протокол IP (Internet Protocol). Этот протокол изначально проектировался как протокол передачи пакетов в составных сетях, объеди-

ненных как локальными, так и глобальными связями. Так как протокол IP является дейтаграммным протоколом, он не гарантирует доставку пакетов до узла назначения. В стеке TCP/IP именно этот протокол и является тем организующим звеном, которое создаёт из набора (разномастных, часто не совместимых друг с другом) сетей сущность более высокого уровня — «сеть сетей», то есть, превращает множество разных объектов в структуру, в «глобальную сеть» (см. определение структуры в [22]).

К уровню межсетевого взаимодействия относятся и все протоколы, связанные с составлением и модификацией таблиц маршрутизации, такие как протоколы сбора маршрутной информации RIP (Routing Internet Protocol) и OSPF (Open Shortest Path First), а также протокол межсетевых управляющих сообщений ICMP (Internet Control Message Protocol). Последний протокол прежде всего предназначен для обмена информацией об ошибках между маршрутизаторами сети и узлом-источником пакета. С помощью специальных пакетов ICMP сообщает о невозможности доставки пакета, о превышении времени жизни или продолжительности сборки пакета из фрагментов, о состоянии системы и т. п.

### 2.1.3. Основной (транспортный) уровень

Поскольку на сетевом уровне не устанавливаются соединения, то нет никаких гарантий, что все пакеты будут доставлены целыми и невредимыми или придут в том же порядке, в котором они были отправлены. Эту задачу решает *основной уровень* стека TCP/IP, называемый также *транспортным*.

На этом уровне функционируют протокол управления передачей TCP (Transmission Control Protocol) и протокол дейтаграмм пользователя UDP (User Datagram Protocol). Протокол TCP обеспечивает надежную передачу сообщений за счет *образования* логических соединений. Этот протокол позволяет равноранговым объектам (объектам одного уровня) поддерживать обмен данными в дуплексном режиме. Он позволяет без ошибок доставить сформированный поток байт в любой другой компьютер, входящий в составную сеть. На передающем узле протокол TCP делит поток байт на части — *сегменты* и передает их ниже лежащему уровню межсетевого взаимодействия. После того как эти сегменты прибудут в пункт назначения, протокол TCP на приёмном узле снова соберет их в непрерывный поток байт.

Протокол UDP обеспечивает передачу прикладных пакетов дейтаграммным способом, как и протокол IP, и выполняет только функции связующего звена (мультиплексора) между сетевым протоколом и многочисленными сервисами прикладного уровня или пользовательскими процессами.

### 2.1.4. Прикладной уровень

Этот объединяет все сервисы, предоставляемые системой пользовательским приложениям. Прикладной уровень реализуется программными системами, построенными в архитектуре клиент-сервер. В отличие от протоколов остальных трех уровней, протоколы прикладного уровня занимаются деталями конкретного приложения и «не интересуются» способами передачи данных по сети. Этот уровень до сих пор постоянно расширяется за счет присоединения к старым, прошедшим многолетнюю эксплуатацию сетевым сервисам типа Telnet, FTP, TFTP, DNS, SNMP сравнительно новых сервисов таких, например, как протокол передачи гипертекстовой информации HTTP, torrent, SIP и т. д..

### 2.1.5. Уровень сетевых интерфейсов

Протоколы этого уровня должны обеспечивать интеграцию в составную сеть (в «сеть сетей») других сетей, причем сеть TCP/IP (эта самая «сеть сетей») должна иметь средства включения в себя любой другой сети, какую бы внутреннюю технологию эта сеть не использовала. Отсюда следует, что этот уровень нельзя определить раз и навсегда поскольку новые сетевые технологии постоянно появляются — научно-технический прогресс на месте не стоит.

Поэтому уровень сетевых интерфейсов в стеке TCP/IP не регламентируется, но он должен поддерживать все популярные стандарты физического и канального уровней как существующие, так и вновь разрабатываемые (локальные сети Ethernet (RFC 894), Token Ring, FDDI, Fast Ethernet, Gigabit Ethernet, 100VG-AnyLAN, глобальные сети — протоколы соединений «точка-точка» SLIP, PPP, HDLC, протоколы территориальных сетей с коммутацией пакетов X.25, frame relay, ATM (RFC 1932) и т. д.). И даже голуби («голубиная почта») могут использоваться в качестве транспорта канального уровня (RFC 1149, RFC 2549, RFC 6214).

### 2.1.6. Стек TCP/IP с точки зрения ЭМВОС

Так как стек TCP/IP был разработан до появления модели взаимодействия открытых систем ISO/OSI, то, хотя он также имеет многоуровневую структуру, соответствие уровней стека TCP/IP уровням модели OSI достаточно условно (см. рис. 17 и 18), но в тоже время не противоречит им.

| Уровни модели OSI | | | | | | | Уровни стека TCP/IP |
|---|---|---|---|---|---|---|---|
| 7 | WWW, Gopher, WAIS | SNMP | FTP | telnet | SMTP | TFTP | I |
| 6 | | | | | | | |
| 5 | TCP | | | | | UDP | II |
| 4 | | | | | | | |
| 3 | IP | ICMP | RIP | OSPF | | ARP | III |
| 2 | Не регламентируется<br>Ethernet, Token Ring, FDDI, X.25, SLIP, PPP, голуби и др. | | | | | | IV |
| 1 | | | | | | | |

*Рис. 17.* Соответствие уровней стека TCP/IP семиуровневой модели OSI. На рисунке показаны далеко не все протоколы стека

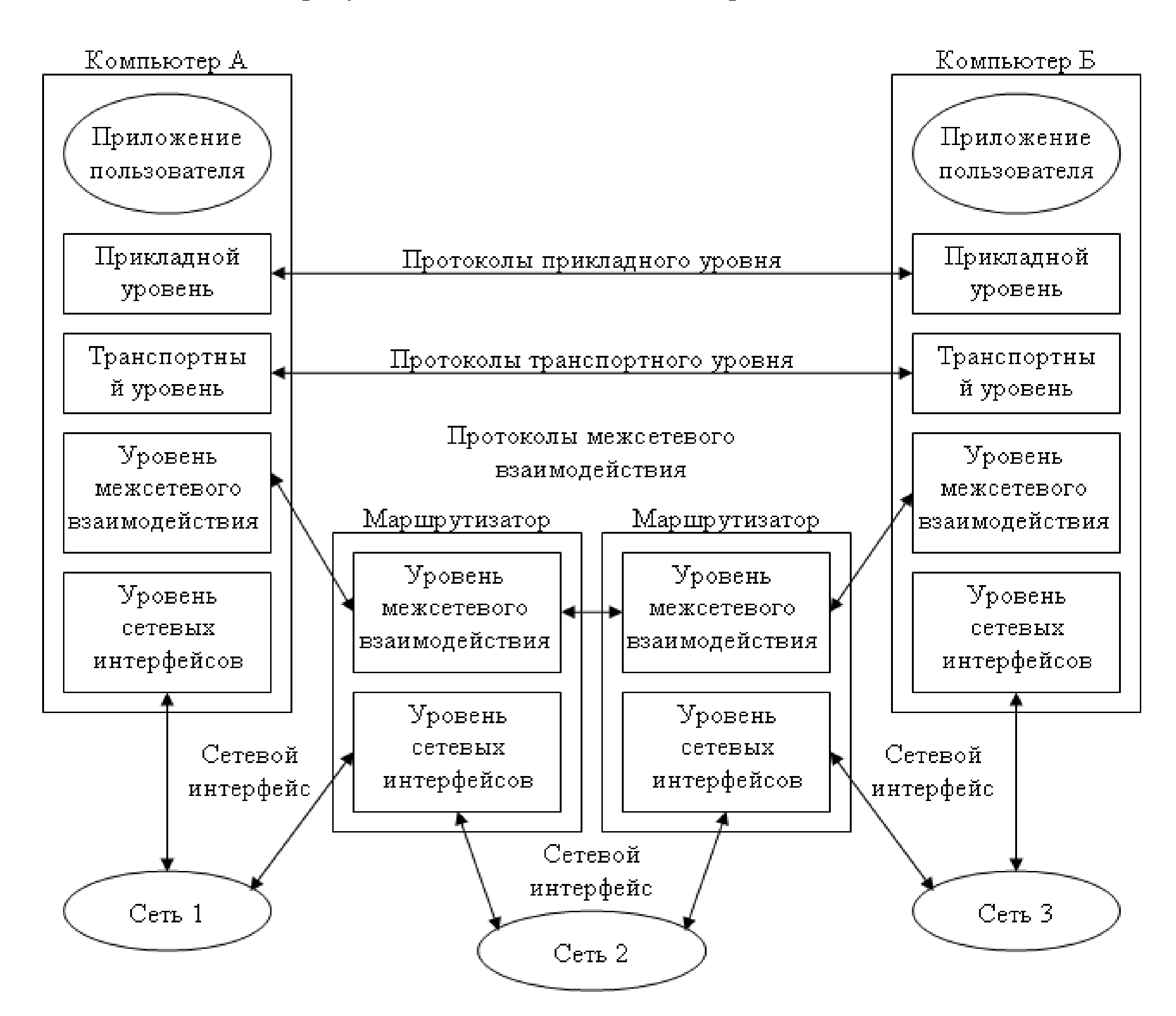



*Рис. 18.* Сетезависимые и сетенезависимые уровни стека TCP/IP. Широкие серые стрелки определяют логические взаимодействия (логическую топологию). Чёрные тонкие стрелки — физические взаимодействия (физическую топологию)

Каждый коммуникационный протокол стека оперирует с некоторой единицей передаваемых данных. Названия этих единиц иногда закрепляются стандартом, а чаще просто определяются традицией. В стеке TCP/IP за многие годы его существования образовалась устоявшаяся терминология в этой области (рис. 19).

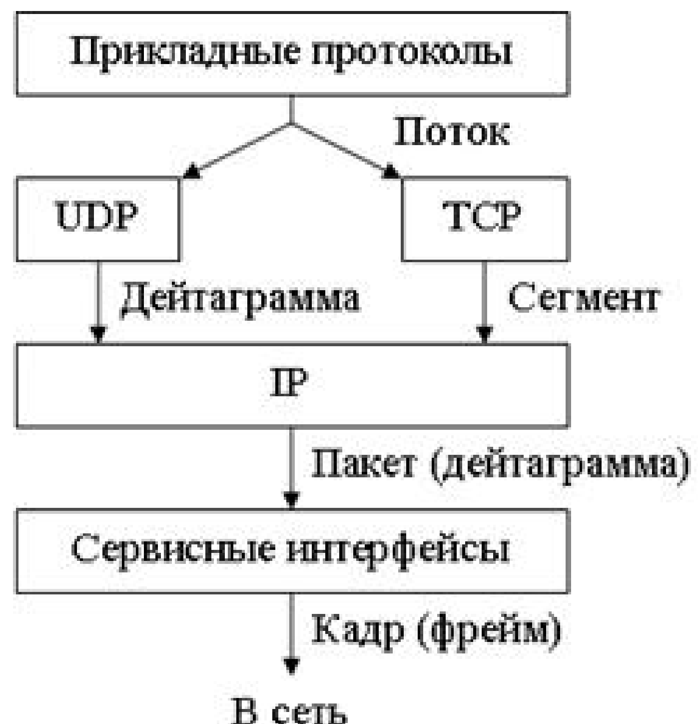



*Рис. 19.* Название единиц данных, используемые в TCP/IP

*Потоком* называют данные, поступающие от приложений на вход протоколов транспортного уровня TCP и UDP.

Протокол TCP нарезает из потока данных *сегменты*.

Единицу данных протокола UDP часто называют *дейтаграммой* (или датаграммой). Дейтаграмма — это общее название для единиц данных, которыми оперируют протоколы без установления соединений. К таким протоколам относится и протокол межсетевого взаимодействия IP.

Дейтаграмму протокола IP называют также *пакетом*.

В стеке TCP/IP принято называть *кадрами (фреймами)* единицы данных протоколов, на основе которых IP-пакеты переносятся через подсети составной сети. При этом не имеет значения, какое название используется для этой единицы данных в конкретной сетевой технологии. Например, в сетевой технологии Ethernet эти единицы данных также называются кадрами, но в других сетевых технологиях они могут иметь иное название.

## 2.2. Определение локальной сети в стеке TCP/IP

Определения локальной сети, которые вы найдёте в Интернет. Разбираем смысл на контрпримерах:

а) локальная сеть — это:

- высокая скорость передачи информации (не менее 10 Мбит/с);

а, вот, у нас в лаборатории 100 Мбит, и то только потому, что коммутатор 100 Мбитный, а был бы 1 Гбитный, то была бы 1-Гбитная сеть; это высокая скорость или невысокая? А у себя дома вы на какой скорости выходите в Интернет, какую скорость вам провайдер обеспечивает, 100 Мбит? А соединение компов по WiFi со скоростью до 150 Мбит (стандарт 802.11n)? То есть, это всё локальные сети?

б) локальная сеть — это:

- низкий уровень ошибок передачи (высококачественные каналы связи) — допустимая вероятность ошибок передачи данных — 10 в 8 степени;

а вы когда последний раз сталкивались с ошибками при работе в Инете? Можете вспомнить — было такое?

в) локальная сеть — это:

- регламентированное количество компьютеров, подключаемых к сети;

а, вот, у некоторых провайдеров локальные сети строятся на основе 10-ой сетки, а в ней 16 млн адресов; это много или мало? Пусть мы смотрим в день по 100 компов в поисках нужного нам файла , тогда мы потратим на это 160000 дней, или 450 лет; это большая сеть или маленькая? Это верхний предел ограничений для локальной сети. А с другой стороны, минимальная сетка адресов, что может быть создана и использована — это два адреса: сеть с маской 255.255.255.252.

г) локальная сеть — это:

- близко расположенные компы;

вы с ноутбуком находитесь в лаборатории; пусть на вашем ноутбуке запущен сервис ftp, web-сервер, треккер torent`а, на веб-сервере (ноутбука) вы выложили видео со своей камеры (ноутбука) и т. д.; а с компа лаборатории (с обычного компа, подключенного к универской кабельной системе) вы подключаетесь к своему ноутбуку и работаете с ним; это близко или далеко? А то, что при этом инфа идёт как минимум через Москву (а может даже так: УлГУ-Москва-Хельсинки-Стокгольм-Франкфурт-на-Майне-Москва-УлГУ) — вас не смущает?

д) локальная сеть — это:

- локальные сети используются для разделения (совместного использования) таких ресурсов, как дисковое пространство;
а как тогда понимать dropbox, yandex-disk, google-disk, а сейчас еще и облачные хранилища, доступные, например, одновременно с компа лаборатории и с вашего смартфона?

е) локальная сеть — это:

- локальные сети используются для разделения (совместного использования) таких ресурсов, как принтеры;
но что вам мешает расшарить свой принтер дома в Инет и печатать на нём вот, прям, отсюда; например, загляните в настройки chromium`a, chromium это поддерживает;

ж) локальная сеть — это:

- локальные сети позволяют осуществлять обмен информацией между компьютерами разных типов;
а в Инете не так? в Инете все компы точно одинаковые?

з) локальная сеть — это:

- локальные сети дают возможность организовать многомашинную вычислительную среду на всех компьютерах сети, что ускоряет решение сложных, ресурсоемких задач;
а проект поиска внеземных цивилизаций SETI — это не то? А взлом алгоритма шифрования MD5 с помощью виртуального кластера — не то? А сервис облачных вычислений тоже не подходит под это определение?

и) локальная сеть — это:

- в локальной сети можно управлять работой технологической системы или исследовательской установкой в режиме реального времени с нескольких компьютеров одновременно;

- так вот же у нас в лаборатории кластер стоит, который настраивался нашим студентом Ахметовым Д.М. из дома по виртуальному тоннелю, проложенному сквозь прокси и fw — это была его дипломная работа [25, 26]. А другой наш студент Кудряшов А.В. в своей дипломной работе реализовал многопользовательскую систему управления роботами через Интернет [23, 24].

Так что же такое локальная сеть?

Правильное определение было дано в пункте 1.5.3 и дополнительное разъяснение в пункте 1.7.6. Прочтите их ещё раз.

## 2.3. Корпоративная сеть в стеке TCP/IP

Как много написано в Интернет о корпоративных сетях! Даже ленивые о них пишут.

Но попробуйте ответить на вопрос:

- а почему именно корпоративная сеть необходима организациям? Чем локальная сеть не устраивает?

Ведь, управление ресурсами, распределение информации, клиент-серверный режим работы, расшаривание приложений корпоративного уровня и прочее — в локальной сети решается не просто проще, а намного проще.

Вас смущает недостаточный «диаметр сети»? Действительно, при использовании сетевых технологий в классическом варианте возникают существенные ограничения на диаметр сети — ограничения кабельной системы. Но в настоящее время этот вопрос решается использованием а) коммутаторов, снимающих ряд ограничений и б) использованием оптоволокна. В совокупности, использование коммутаторов с оптоволоконными каналами для взаимосвязи между собой позволяет увеличить диаметр локальной сети до многих-многих километров и десятков километров. То есть, уже нет ограничений на использование технологий локальных сетей для обеспечения/решения задач корпоративного уровня.

Использование локальных сетей для решения задач управления корпоративными ресурсами, распределение информации в организации, клиент-серверный режим работы, расшаривание корпоративных приложений и т. д. — и проще технологически, и в разы дешевле. То есть, и оборудование проще, и, соответственно, кадровый вопрос решается проще.

И тем не менее организации от локальных сетей переходят к корпоративным. Почему?

Ответ в Интернет вы найдёте с превеликим трудом. Если вообще найдёте.

**Правильный ответ:** обеспечение разграничения доступа к информации. Именно это является решающим фактором для принятия решения о переходе от локальной сети организации к корпоративной.

**Пояснение.** В локальной сети разграничения доступа базируется на защите компьютера и защите самого расшаренного в сеть приложения. То есть, защита информации — локальна и этого оказывается совершенно недостаточно для полноценной защиты информации. И вопрос защиты ещё более обостряется, если учесть, что в большой организации некоторый бизнес-процесс (бизнес-функцию), как правило обеспечивает некоторый

коллектив людей — подразделение. То есть, некоторое количество компьютеров в сети используются для реализации некоторого бизнес-процесса. В локальной сети все компы видны. И следовательно, несложно «подсмотреть», чем же там люди занимаются, поскольку трафик доступен и его несложно расшифровать.

Выход — сегментация локальной сети, деление её на подсети, установка маршрутизаторов для организации связи между подсетями, установка на маршрутизаторах фильтрации трафика. Тем самым, защита информации становится двухуровневой, и не просто двухуровневой, а ещё и технологически разной, ибо защита маршрутизаторов основывается на несколько иных принципах, нежели локальная защита компов и приложений. Иначе говоря, требования к квалификации взломщика выводятся на профессиональный уровень, а подобные умения уже в кармане не спрячешь.

Все остальные пункты требований к корпоративной сети практически ничем не отличаются от аналогичных требований к локальной сети, более того, на уровне локальной сети они реализуются проще и, соответственно, в сопровождении оказываются дешевле.

Таким образом, именно требование обеспечения разграничения доступа к информации является определяющим для перехода от использования локальных сетей к корпоративным.

## 2.4. Местоположение стека TCP/IP в системе

Уровень сетевых интерфейсов в настоящее время не входит в состав стека и реализуется в основном аппаратно, например, в сетевых картах, а интерфейс к ним реализуется программно в драйверах сетевых карт.

Уровни межсетевого взаимодействия и транспортный располагаются непосредственно в ядре ОС. Это обусловлено необходимостью максимально быстро обрабатывать сетевые пакеты проходящие от прикладных программ к уровню сетевых интерфейсов и обратно (см. рис. 20). Три стрелки сверху на рисунке показывают вход/выход в сетевую подсистему со стороны API ОС, то есть, в составе API ОС есть некоторый набор системных вызовов, с помощью которых прикладное ПО может взаимодействовать с сетью.

Таким образом, стек TCP/IP в ядре ОС реализуется с помощью модулей:

1. Драйверы сетевых устройств; подразумевается наличие одного драйвера для каждого устройства.

2. Аппаратно-независимый интерфейс, предоставляющий постоянный (одинаковый) вид всех устройств, так что всем вышестоящим модулям не нужно знать особенности используемого оборудования.

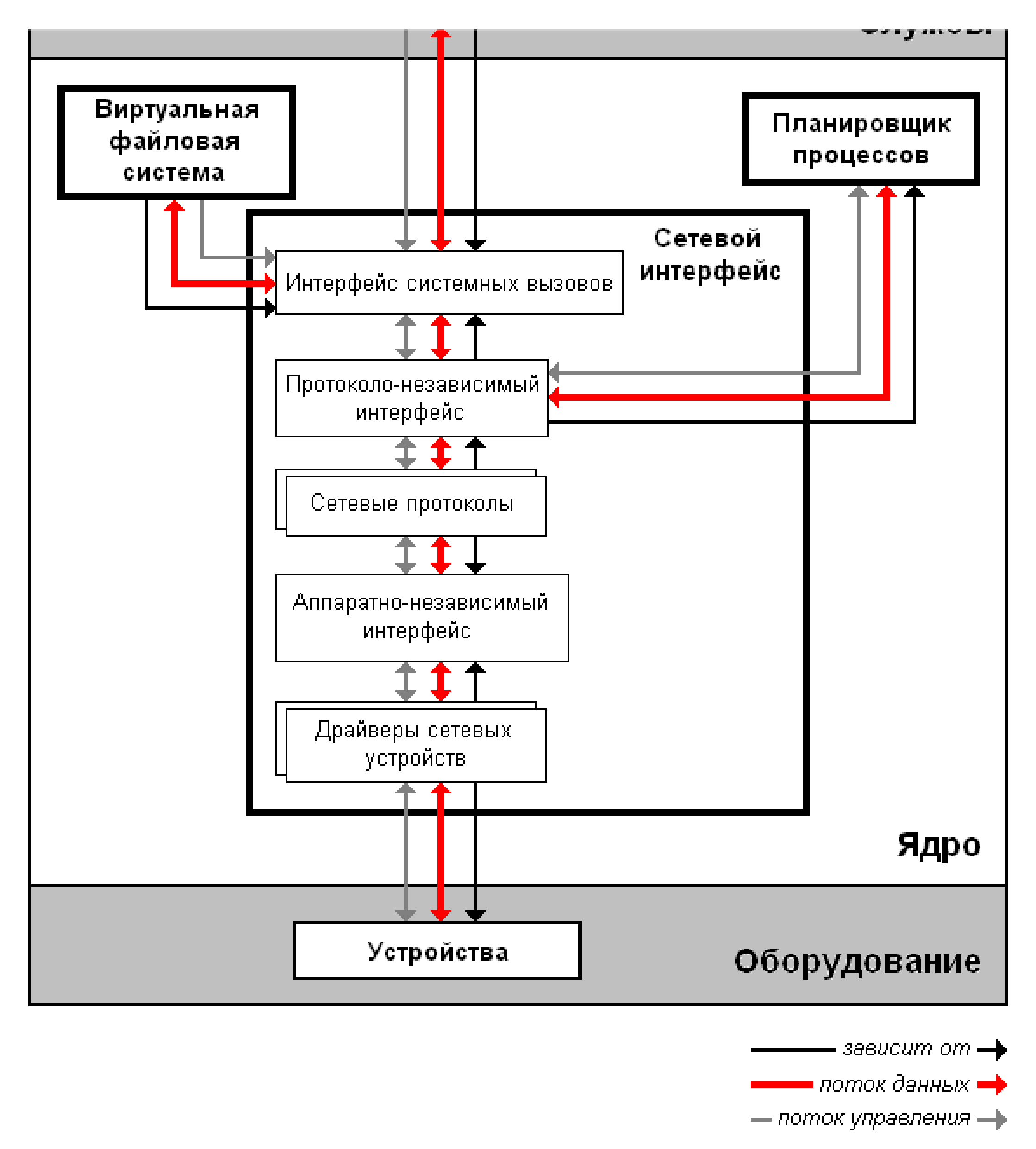



*Рис. 20.* «Сетевые протоколы» — протоколы сетевого и транспортного уровня стека TCP/IP

3. Сетевые протоколы, реализующие протоколы уровней межсетевого взаимодействия (IP, ICMP, OSPF и др.) и транспортного (TCP, UDP и др.)

4. Интерфейсный модуль, абстрагирующий всё нижестоящее так, что ни другим подсистемам ядра, ни, тем более, прикладному ПО не нужно знать, какие конкретно устройства и протоколы используются для взаимодействия.

5. Некоторые функции интерфейсного модуля экспортируются (объявляются всем видимыми, то есть, включаются в API ОС как системные вызовы), чтобы пользовательские процессы могли иметь к ним доступ.

Уровень прикладной стека TCP/IP (прикладные сетевые протоколы) располагается вне ядра ОС («сверху» ОС) и реализуется, как правило, в виде «расшариваемых» библиотек (обычно говорят «динамически компонуемых») или даже, если прикладной протокол «узкоспециализированный», то реализуется в самом приложении, как набор функций.

Сетевое взаимодействие реализуется с помощью механизма сокетов (socket), создаваемых системным вызовом socket() (см. API ОС). Сокеты ассоциируются (связываются, см. именование на верхних уровнях стека) с пользовательскими процессами, в том числе, могут использоваться совместно несколькими процессами, если в структурах этих процессов есть указатели на структуру одного и того же сокета.

**Вопросы «на засыпку»**

1. Могу ли я, частное лицо — Чичев Александр Алексеевич, преподаватель УлГУ, создать и соответственно владеть/администрировать корпоративной сетью?

2. А можете ли вы, частное лицо — студент имярек, группа *вписать нужную*, создать и соответственно владеть/администрировать корпоративной сетью?

3. А может ли моя соседка по дому, по лестничной площадке создать и соответственно владеть/администрировать корпоративной сетью?

4. А будет ли она мешать мне (с технической точки зрения) создавать и соответственно владеть/администрировать моей корпоративной сетью?

5. Какой статус должно иметь частное лицо, чтобы создать и соответственно владеть корпоративной сетью?

6. А какой статус должна иметь фирма/организация/контора, чтобы создать и соответственно владеть корпоративной сетью?

7. А если только локальной сетью?

8*. Каким образом один и тот же сокет может использоваться совместно несколькими процессами?

# Лекция 3. Стек TCP/IP. Протокол IP

## 3.1. Описание протокола IP

### 3.1.1. Назначение протокола

Протокол IP это основной механизм стека TCP/IP, обеспечивающий создание «сети сетей», то есть, объединение всех нижестоящих сетей (независимо от их внутреннего устройства) в единую конструкцию — объединённую сеть.

В рамках этой цели протокол IP предназначен для реализации двух основных функций:

- обеспечить логическую адресацию (именование) устройств в объединённой вычислительной сети; причём адрес (имя) каждого устройства, «видимого» в сети, должен быть уникальным в пределах всей объединённой сети;

- предоставить услугу вышестоящему транспортному уровню в пересылке (передаче) данных между сетевыми узлами; то есть, вышестоящие протоколы TCP и UDP рассматривают IP как рабочую лошадку, которая таскает пакеты из точки А в точку Б, сквозь множество различных промежуточных подсетей.

*Определение 1-ой функции:* Почему обеспечивается именование логическое? Это объяснялось в предыдущей лекции: на физическом и канальном уровнях ЭМВОС (где действуют физические адреса устройств) имеют место быть весьма жёсткие ограничения, обусловленные свойствами среды и оборудования, преодолеть которые не представляется возможным и которые в итоге приводят к появлению такого понятия, как «диаметр сети». Следствием этой ситуации является то, для создания более крупных конструкций (то есть, объединения сетей) приходится задействовать более сложные логические методы.

*Определение 2-ой функции:* Поскольку возможных путей доступа из точки А в точку Б в большой сети может быть несколько, то для эффективного выполнения функции пересылки данных протокол должен уметь выбирать путь (маршрут) сквозь промежуточные узлы/сети. Для поддержки этого «умения» на сетевом уровне работают протоколы динамической маршрутизации OSPF, RIP, EIGRP (Routing protocols — маршрутизирующие протоколы, то есть те, кто маршрутизирует), которые рассчитывают и

учитывают возможные пути доступа из точки А в точку Б, включая промежуточные точки на маршрутах. Этой информацией они делятся с протоколом IP, который в силу этого называется Routed, то есть, маршрутизируемым. В помощь им, для решения непредвиденных ситуаций, используется протокол ICMP (Internet Control Message Protocol), который также относится к сетевому уровню

### 3.1.2. Формат пакета протокола

**IP-пакет —** форматированный блок информации, передаваемый по компьютерной сети, структура которого определена протоколом IP.

**Версия IPv4.** В современной сети Интернет используется IP четвёртой версии (рис. 21), также известный как IPv4. В протоколе IP этой версии каждому узлу сети ставится в соответствие IP-адрес длиной 4 октета (4 байта). При этом компьютеры в подсетях объединяются общими начальными битами адреса. Количество этих бит, общее для данной подсети, называется маской подсети (ранее использовалось деление пространства адресов по классам — A, B, C; класс сети определялся диапазоном значений старшего октета и определял число адресуемых узлов в данной сети, сейчас используется бесклассовая адресация).

- **Версия** — для IPv4 значение поля должно быть равно 4.

- **IHL** — (Internet Header Length) длина заголовка IP-пакета в 32-битных словах (dword). Именно это поле указывает на начало блока данных (*payload* — полезный груз) в пакете. Минимальное корректное значение для этого поля равно 5.

- **Длина пакета** — (Total Length) длина пакета в октетах, включая заголовок и данные. Минимальное корректное значение для этого поля равно 20, максимальное — 65 535 (64 кб).

- **Идентификатор** — (Identification) значение, назначаемое отправителем пакета и предназначенное для определения корректной последовательности фрагментов при сборке пакета. Для фрагментированного пакета все фрагменты имеют одинаковый идентификатор.

- **3 бита флагов**. Первый бит должен быть всегда равен нулю, второй бит DF (don’t fragment) определяет возможность фрагментации пакета и третий бит MF (more fragments) показывает, не является ли этот пакет последним в цепочке пакетов.

- **Смещение фрагмента** — (Fragment Offset) значение, определяющее позицию фрагмента в потоке данных. Смещение задается количеством восьмибайтовых блоков, поэтому это значение требует умножения на 8 для перевода в байты.

- **Время жизни** (TTL) — число маршрутизаторов, которые может пройти этот пакет. При прохождении маршрутизатора это число уменьшается на единицу. Если значение этого поля равно нулю, то пакет должен быть отброшен, и отправителю пакета может быть послано сообщение *Time Exceeded* (протокол ICMP, тип 11, код 0).

- **Протокол** — идентификатор сетевого протокола следующего (вышестоящего) уровня указывает, данные какого протокола содержит пакет, например, TCP, UDP, или ICMP (см. IANA protocol numbers и RFC 1700). В IPv6 называется «Next Header».

- **Контрольная сумма заголовка** — (Header Checksum) вычисляется в соответствии с RFC 1071

**Версия 6 (IPv6)**. С 1996 года вводится в эксплуатацию шестая версия протокола — IPv6 (рис. 22), которая позволяет адресовать значительно большее количество узлов, чем IPv4. Адресное пространство IPv6 составляет $2^{128}$. Такое большое адресное пространство было введено ради иерархичности адресов (это упрощает маршрутизацию). Тем не менее, увеличенное пространство адресов сделает NAT необязательным. Классическое применение IPv6 (по сети /64 на абонента; используется только unicast-адресация) обеспечит возможность использования более 300 млн IP-адресов на каждого жителя Земли. Эта версия отличается повышенной разрядностью адреса, встроенной возможностью шифрования и некоторыми другими особенностями. Долгий переход с IPv4 на IPv6 связан с трудоёмкой работой операторов связи и производителей программного обеспечения и не может быть выполнен в один момент.

- **Версия** — для IPv6 значение поля должно быть равно 6.

- **Класс трафика** — определяет приоритет трафика (QoS, класс обслуживания).

- **Метка потока** — уникальное число, одинаковое для однородного потока пакетов.

- **Длина полезной нагрузки** — длина данных в октетах (заголовок IP-пакета не учитывается).

- **Следующий заголовок** — задаёт тип расширенного заголовка (*IPv6 extension*), который идёт следующим. В последнем расширенном заголовке поле *Next header* задаёт тип транспортного протокола (TCP, UDP и т. д.) и определяет следующий инкапсулированный уровень.

- **Число переходов** — максимальное число маршрутизаторов, которые может пройти пакет. При прохождении маршрутизатора это значение уменьшается на единицу и по достижении нуля пакет отбрасывается.

<table>
<tr><th>Октет</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td colspan="4">Версия</td><td colspan="4">IHL</td><td colspan="6">Differentiated Services Code Point</td><td colspan="2">ECN</td><td colspan="16">Длина пакета</td></tr>
<tr><td>4</td><td colspan="16">Идентификатор</td><td colspan="2">Флаги</td><td colspan="14">Смещение фрагмента</td></tr>
<tr><td>8</td><td colspan="8">Время жизни (TTL)</td><td colspan="8">Протокол</td><td colspan="16">Контрольная сумма заголовка</td></tr>
<tr><td>12</td><td colspan="32">IP-адрес отправителя</td></tr>
<tr><td>16</td><td colspan="32">IP-адрес получателя</td></tr>
<tr><td>20</td><td colspan="32">Параметры (от 0 до 10 32-битных слов)</td></tr>
<tr><td></td><td colspan="32">Данные</td></tr>
</table>

*Рис. 21.* Формат протокола IPv4

<table>
<tr><th>Позиция в октетах</th><th></th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th></th><th>Позиция в битах</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Версия</td><td colspan="8">Класс трафика</td><td colspan="20">Метка потока</td></tr>
<tr><td>4</td><td>32</td><td colspan="16">Длина полезной нагрузки</td><td colspan="8">След. заголовок</td><td colspan="8">Число переходов</td></tr>
<tr><td>8</td><td>64</td><td colspan="32" rowspan="4">IP-адрес отправителя</td></tr>
<tr><td>12</td><td>96</td></tr>
<tr><td>16</td><td>128</td></tr>
<tr><td>20</td><td>160</td></tr>
<tr><td>24</td><td>192</td><td colspan="32" rowspan="4">IP-адрес получателя</td></tr>
<tr><td>28</td><td>224</td></tr>
<tr><td>32</td><td>256</td></tr>
<tr><td>36</td><td>288</td></tr>
<tr><td>40</td><td>320</td><td colspan="32">Данные</td></tr>
</table>

*Рис. 22.* Формат протокола IPv6

## 3.2. Алгоритм протокола

### 3.2.1. Схема алгоритм протокола

IP объединяет сегменты сети (локальные сети) в единую сеть, обеспечивая доставку пакетов данных между любыми узлами сети через произвольное число промежуточных узлов (маршрутизаторов). Протокол IP не гарантирует надёжной доставки пакета до адресата — в частности, пакеты могут прийти не в том порядке, в котором были отправлены, продублироваться (приходят две копии одного пакета), оказаться повреждёнными (обычно повреждённые пакеты уничтожаются) или не прийти вовсе. Гарантию безошибочной доставки пакетов дают некоторые протоколы более высокого уровня (например, TCP), которые используют IP в качестве транспорта.

При доставке IP пакета он проходит через разные каналы доставки. Возможно возникновение ситуации, когда размер пакета превысит возможности узла системы связи. В этом случае протокол предусматривает возможность дробления пакета (фрагментации) на уровне IP в процессе доставки. Соответственно, к конечному получателю пакет придет в виде нескольких пакетов-фрагментов, которые необходимо собрать в один перед дальнейшим анализом. Возможность дробления пакета с последующей сборкой называется IP фрагментацией.

В протоколе предусмотрена возможность запрета фрагментации конкретного пакета. Если такой пакет нельзя передать через сегмент связи целиком, то он уничтожается, а отправителю направляется ICMP сообщение о проблеме.

Программной реализацией алгоритма протокола является модуль протокола, входящего в состав ядра ОС unix/linux (в kernel, в linux — в файле vmlinuz), то есть, модуль протокола содержится в базовой, резидентной части ядра ОС.

Блок-схема IP содержит восемь компонентов (рис. 23):

- модуль, заполняющий заголовок;
- модуль обработки;
- модуль маршрутизации;
- модуль фрагментации;
- модуль реассемблирования;
- таблица маршрутизации;
- таблица MTU;
- таблица реассемблирования.

Кроме того, блок-схема включает в себя исходящие и входящие очереди.

Блок-схема получает пакет либо от звена данных, либо от протокола высокого уровня. Когда пакет поступает от протокола высокого уровня, он доставляется к уровню звена данных для передачи (если это не адрес обратной тестовой петли 127.X.Y.Z). Когда пакет поступает от уровня звена данных, он доставляется либо к уровню звена данных для дальнейшего продвижения (в маршрутизатор), либо на верхний уровень протокола (если этот IP-адрес конечного пункта пакета — такой же, как и адрес станции).

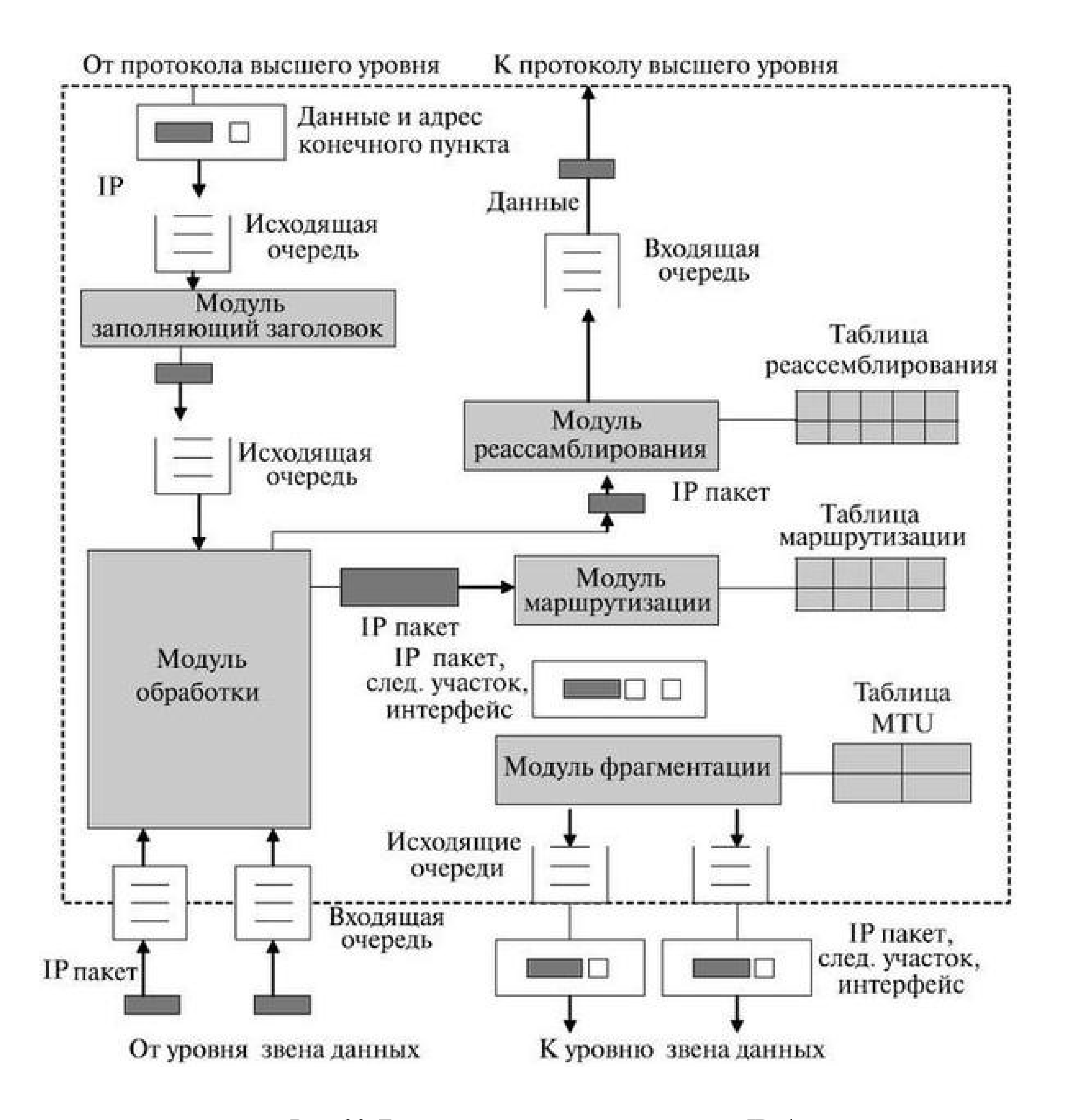



*Рис. 23.* Блок схема алгоритма протокола IPv4

### 3.2.2. Модуль, заполняющий заголовок

Модуль, дополняющий заголовок, получает данные от протокола верхнего уровня наряду с адресом конечного пункта. Он инкапсулирует данные в IP-дейтаграмму, дополняя IP-заголовком (см. рис. 24).




*Рис. 24.* Алгоритм модуля дополняющего заголовок

### 3.2.3. Модуль обработки

Модуль обработки – центральный в совокупности модулей IP (см. рис. 25). Обрабатывающий модуль получает дейтаграмму от интерфейса или от модуля, дополняющего заголовок. В обоих случаях их обрабатывают одинаково. Дейтаграмма должна быть обработана и маршрутизирована независимо от того, откуда она прибыла.

Модуль обработки сначала осуществляет проверку, чтобы определить, является дейтаграмма пакетом тестовой обратной связи с адресом конечного пункта 127.X.Y.Z или пакетом, который достиг своего конечного пункта. В любом случае, пакет посылают модулю реассемблирования.

Если модуль – это маршрутизатор, он уменьшает поле времени жизни (TTL — time to live) на единицу. Если значение поля меньше или равно нулю, дейтаграмма отклоняется и посылается сообщение системы управ-

ления Интернета (ICMP) на начальную станцию. Если значение TTL после уменьшения больше чем нуль, обрабатывающий модуль посылает дейтаграмму к модулю-маршрутизатору.

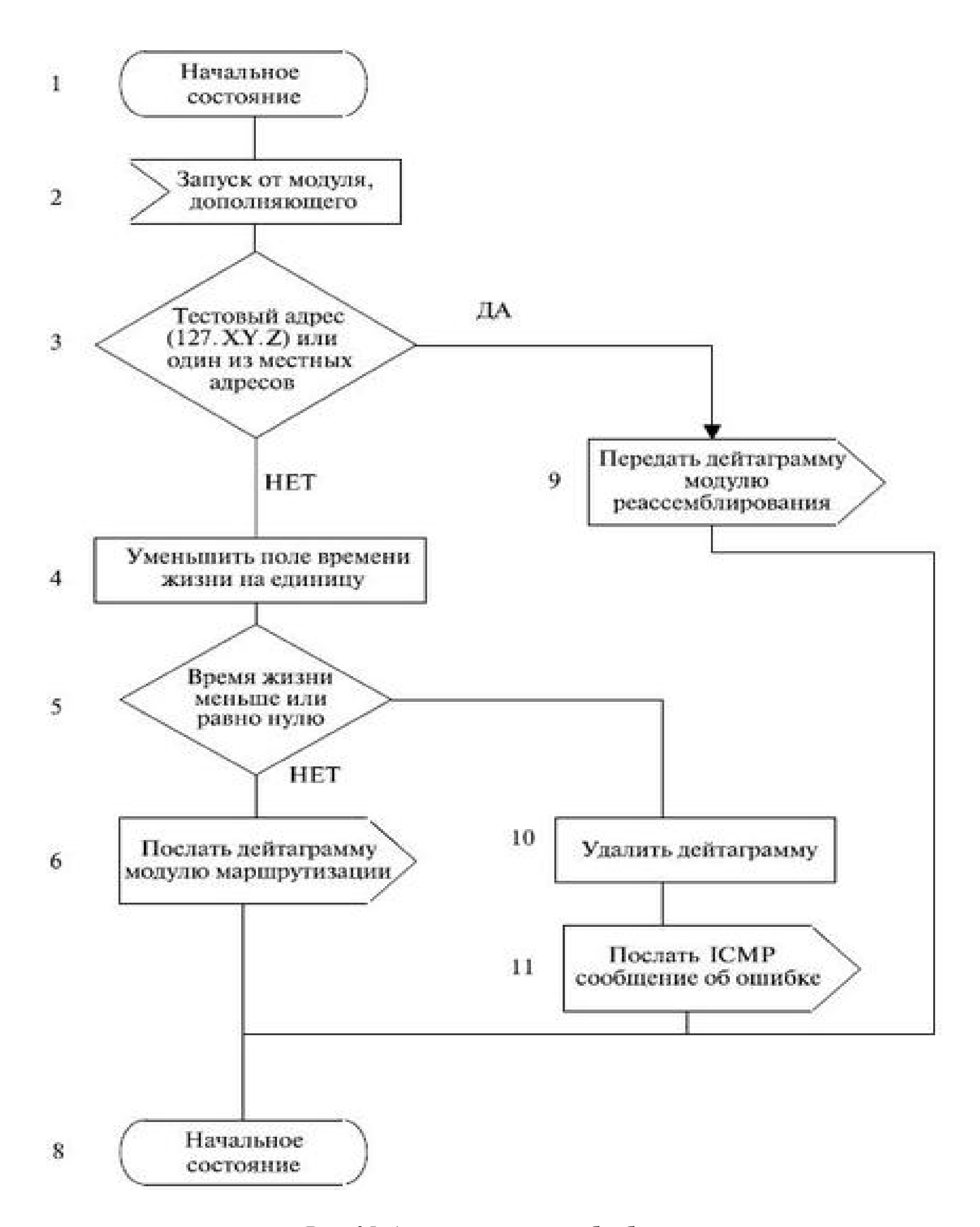



*Рис. 25.* Алгоритм модуля обработки

### 3.2.4. Очереди

Наше пакетирование использует два типа очередей: исходящие очереди и входящие очереди. **Исходящие очереди** накапливают дейтаграммы, приходящие от уровня звена данных или протоколов высшего уровня. **Входящие очереди** накапливают дейтаграммы, идущие к уровню звена данных или протоколов высшего уровня. Модуль обработки выбирает дейтаграммы из исходящих очередей. Модули фрагментации и ассемблирования добавляют дейтаграммы во входящие очереди.

### 3.2.5. Таблица маршрутизации

Таблица маршрутизации используется модулем маршрутизации для определения адреса следующего участка пакета.

### 3.2.6. Модуль маршрутизации

Модуль маршрутизации получает IP-пакет от модуля обработки. Если пакет передается дальше, то это делает этот модуль. Модуль находит IP-адрес следующей станции в соответствии с номером, который должен быть послан в пакете. Затем он посылает пакет с этой информацией в модуль фрагментации.

### 3.2.7. Модуль фрагментации

В нашей блок-схеме модуль фрагментации (см. рис. 26) получает IP-дейтаграммы от модуля маршрутизации. Модуль маршрутизации пересылает: IP-дейтаграмму, IP-адрес следующей станции (либо конечный пункт назначения для прямой доставки, либо следующий маршрутизатор для непрямой доставки) и номер интерфейса, с помощью которого дейтаграмма будет выслана.

Таблица MTU (Maximum Transferred Unit) используется модулем фрагментации, чтобы найти максимальную передаваемую единицу в конкретном интерфейсе.

Модуль фрагментации обращается к таблице MTU, чтобы найти MTU для заданного интерфейса. Если длина дейтаграммы больше чем MTU, модуль фрагментации фрагментирует дейтаграмму, добавляя заголовок к каждому фрагменту, и посылает их в ARP(Address Resolution Protocol) — для определения адреса и доставки.

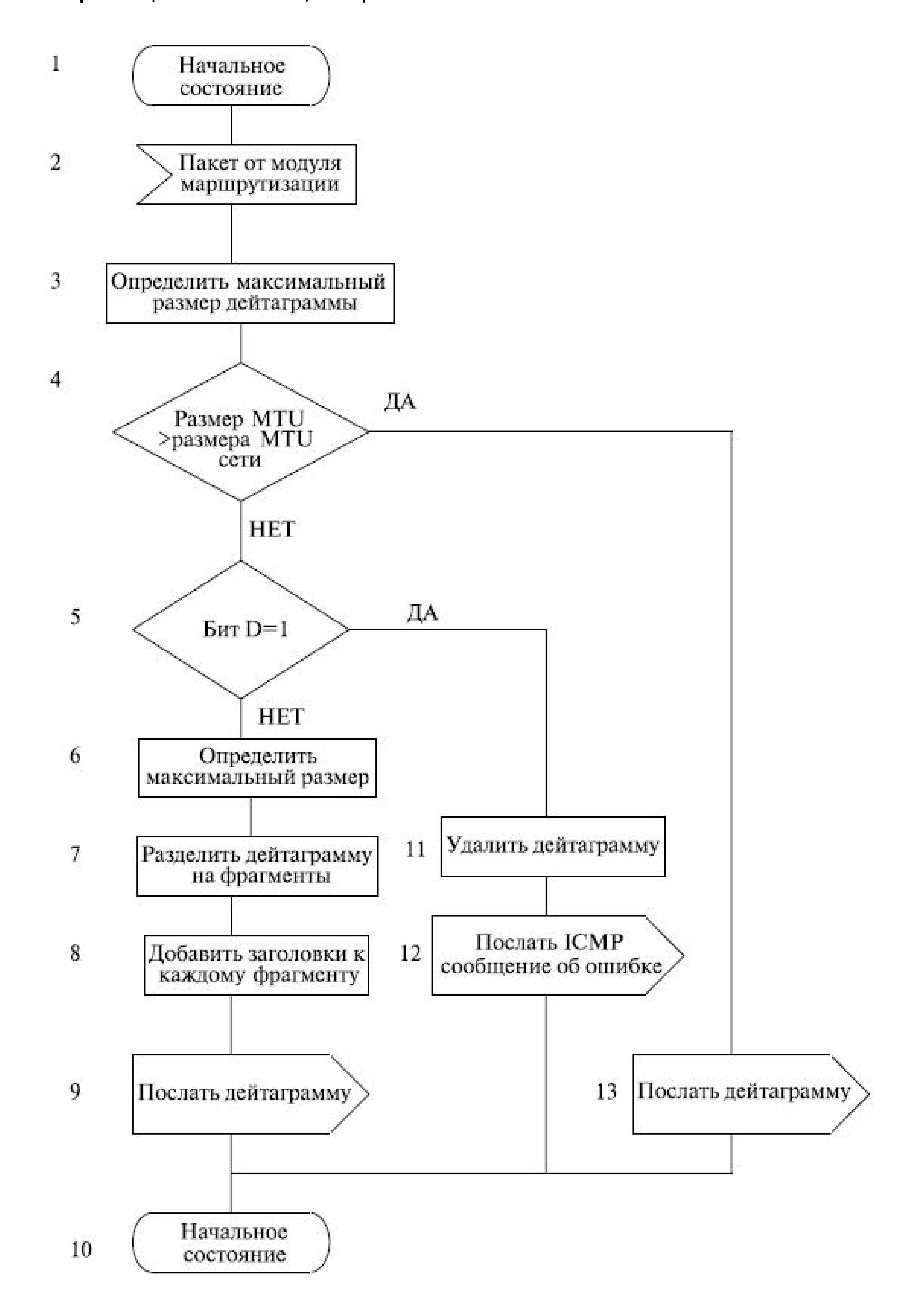



*Рис. 26.* Алгоритм модуля фрагментации

### 3.2.8. Таблица реассемблирования

Таблица реассемблирования используется модулем реассемблирования. В нашем пакетировании таблица реассемблирования имеет пять полей: состояние, адрес источника, IP-дейтаграммы, отсчет времени, ID-дейтаграммы и фрагменты (см. рис. 27).

Значение поля состояние может быть одним из двух: FREE (СВОБОДНО) или IN_USE (в использовании). Поле IP-адреса определяет IP-адреса источника дейтаграммы и все фрагменты, принадлежащие этой дейтаграмме. Отсчет времени – заранее определенное количество времени, в которое каждый фрагмент должен прибыть. В заключение определим: поле "фрагменты" — это указатель списка связи фрагментов.

St. : State
Состояние
S.A.: Source address Адрес источника
T.O.: Time-out Отсчет времени
D.I: DiagramID ID дейтаграммы
F.: Fragment Фрагмент

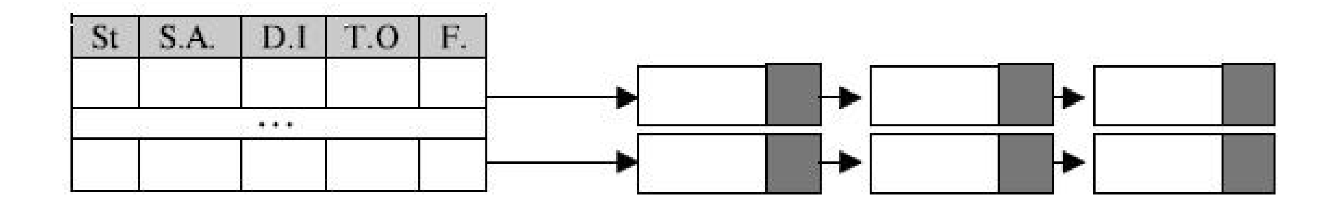



*Рис. 27.* Таблица реассемблирования

### 3.2.9. Модуль реассемблирования

Модуль реассемблирования принимает от модуля обработки те фрагменты, которые прибывают на их конечный пункт. В нашем пакетировании модуль реассемблирования обращается с не фрагментированными дейтаграммами как с фрагментированными, принадлежащими дейтаграмме только с одним фрагментом.

Поскольку IP — протокол без коммутации, нет гарантий, что фрагменты прибывают в порядке. С другой стороны, фрагменты одной дейтаграммы могут быть перемешаны с фрагментами от других дейтаграмм. Для того чтобы зафиксировать такую ситуацию, модуль использует таблицу реассемблирования с вспомогательной таблицей связей, как мы это описывали ранее.

Задача модуля реассемблирования — найти дейтаграмму, которой принадлежат по порядку все фрагменты одной и той же дейтаграммы, и реассемблировать все фрагменты, когда они все прибудут. Если истекает установленный отсчет времени и некоторый фрагмент не соответствует порядку следования, модуль отклоняет фрагмент.

**Вопросы «на засыпку»**

1. Какое поле IP-заголовка меняется от маршрутизатора к маршрутизатору?
2. Дейтаграмма IP должна пройти через маршрутизатор 126.46.10.5. Других ограничений на маршрутизатор нет. Представьте опции с их значениями.
3. Какое максимальное число маршрутизаторов, которое может быть записано, если опция метка времени имеет значение 1? Почему?
4. Может ли значение длины заголовка в пакете IP быть меньше, чем 5? Когда оно точно равно 5?
5. Хост передает 100 дейтаграмм к другому хосту. Если идентификационный номер первой дейтаграммы равен 1024 , каков идентификационный номер последней?

# Лекция 4. Стек TCP/IP. Цифровое именование

## 4.1. Именование в протоколе

### 4.1.1. Именование взаимодействующих объектов в протоколе

Протокол IP работает только с цифровой формой имени, о существовании символьной формы имени протокол не знает от слова совсем и потому преобразование имени должно осуществляться до поступления имени в протокол.

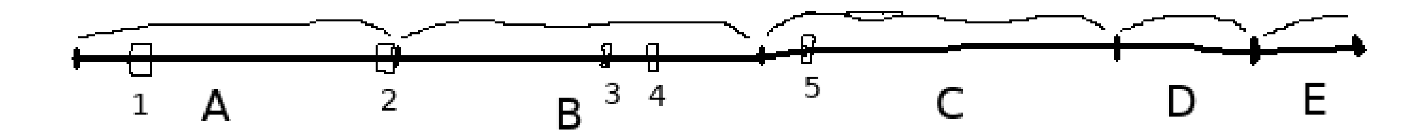
Имя (IP-адрес) узла назначения (узла-получателя) поступает в протокол

- либо сверху — так обычно происходит, если протокол работает в обычном компе пользователя, то есть, узел выполняет роль «Оконечного Оборудования Данных» (ООД, см. п. 1.2.1. рис. 1); на рис. 27 этот вход обозначен как «От протокола высшего уровня»; то есть, сверху спускается отдельно пакет данных в формате вышестоящего протокола и отдельно имя узла назначения и протокол IP формирует свой пакет;

- либо снизу — так обычно происходит, если протокол работает в устройстве-маршрутизаторе, то есть, узел выполняет роль «Промежуточного оборудования сети» (см. рис. 1); на рис. 27 этот вход обозначен как «От уровня звена данных»; то есть, снизу поднимается IP-пакет с адресом получателя в пятом слове.

В пакете протокола адрес (и отправителя, и получателя) — слова 4 и 5, хранятся в 16-ричном виде, не в привычном нам формате с точками (дот-форма), а именно в 16-ричном виде. Ну, или, если побитово рассматривать слово протокола, то в двоичном. И причём, не просто в 16-ричном (двоичном), а в «сетевом» формате байт — младшие биты/байты идут первыми. Напоминаем инфу из дисциплины «Архитектура ЭВМ»: в интеловских процессорах определён несколько иной порядок байт и нельзя просто так «запихивать» 16-ричный адрес в пакет протокола, адрес нужно преобразовать в сетевой порядок байт.

Пространство цифровых имён сетевых объектов, используемых протоколом, имеет линейную структуру — плоское пространство имён (см. рис. 28).




Приватные сетки:

1 - 10.0.0.0
4 - 172.16.0.0 - 172.32.0.0
5 - 192.168.0.0

Дополнительная приватная
для Zeroconf (avahi):
3 - 169.154.0.0

Сетка для интерфейса lo0:

2 - 127.0.0.0

*Рис. 28.* Пространство цифровых имён протокола IP. Крайний слева адрес — 0.0.0.0 (0x0), а крайний справа адрес — 255.255.255.255 (0xFFFFFFFF)

При создании протокола было определено, что это пространство имён разбивается на классы сеток A, B, C, D, E с границами, как показано на рисунке 29. На этом рисунке показан формат цифрового имени в разных классах сеток, а в таблице 1 — характеристики классов.

**Таблица 1. Характеристики классов**

| **Класс** | **Первые биты** | **Наименьший адрес (номер) сети** | **Наибольший адрес (номер) сети** | **Маска сети** | **Максимальное число подсеток в одной сети** | **Максимальное число узлов в сети** |
|---|---|---|---|---|---|---|
| A | 0 | 1.0.0.0 | 126.0.0.0 | 255.0.0.0 | 126 | 2**24 — 2 |
| B | 10 | 128.0.0.0 | 191.255.0.0 | 255.255.0.0 | 16 384 | 2**16 — 2 |
| C | 110 | 192.0.0.0 | 223.255.255.0 | 255.255.255.0 | 2 097 152 | 2**8 — 2 |
| D | 1110 | 224.0.0.0 | 239.255.255.255 | | | Multicast |
| E | 11110 | 240.0.0.0 | 255.255.255.255 | | | зарезервиро-ван |

Разработчиками предполагалось, что большие сетки класса A будут использовать крупные организации/фирмы/корпорации, сетки класса B — организации/фирмы поменьше и т. д. Однако, действительность сущест-

венно превзошла ожидания и до сих пор не появилось ни одной фирмы/корпорации, которая смогла бы задействовать адресное пространство сетки класса А хотя бы процентов на 10.

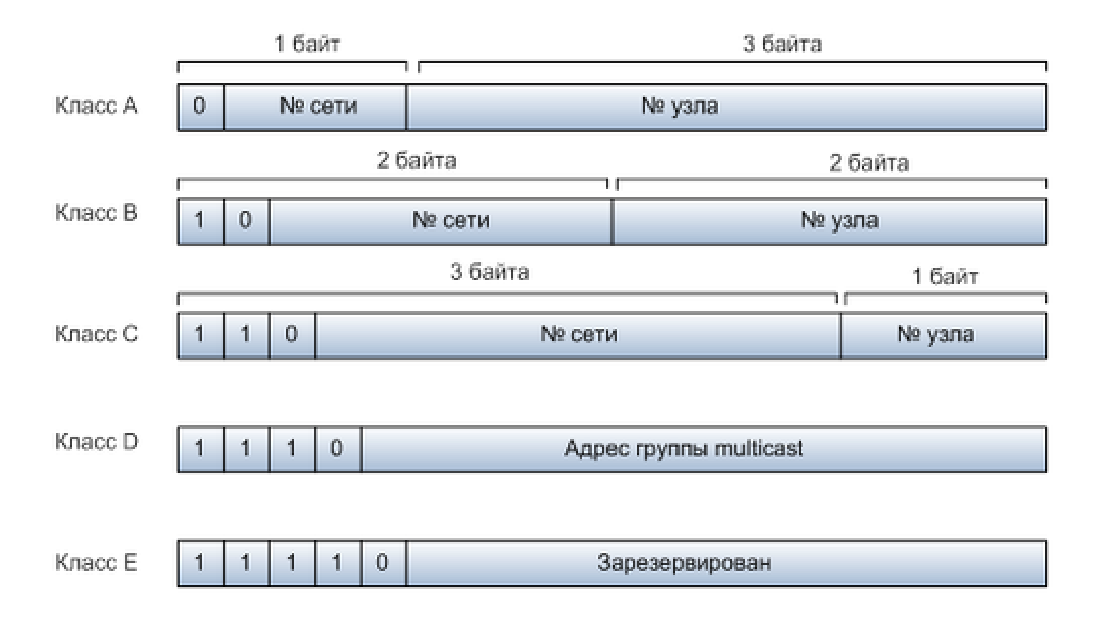



*Рис. 29.* Формат цифровых имён в разных классах

А уже в 90-е годы, вскоре после того как Интернет стал международным, выяснилось, что таким образом определённое пространство цифровых имён совершенно недостаточно и пришлось перейти к бесклассовой адресации CIDR (Classless Inter-Domain Routing — RFC2050), которая была сложнее в эксплуатации, но позволяла более экономно использовать адресное пространство IPv4. Также дополнительно в этих же целях (экономии) стал широко использоваться протокол NAT (трансляция сетевых адресов, RFC 1631, RFC 3022, RFC 6886), когда небольшое количество адресов, соответствующих требованиям IANA («белых», очень часто только один «белый» адрес), используются большим пулом частных адресов, не маршрутизируемых глобально и топологически находящихся за блоком NAT. Поскольку более радикальное решение – переход на IPv6, оказалось невозможно быстро осуществить по технологическим причинам. Также как в классовой адресации, в бесклассовой принята маска в виде непрерывной последовательности единиц и непрерывной последовательности нулей (см. Таблицу 1). Только для таких масок получающиеся множества IP-адресов будут смежными. Однако также широко распространены обратные маски (invers mask, wildcard mask), которые не обязаны содержать подряд идущие единицы или нули. Однако их применение, как правило, ограничивается лишь формированием правил ACL.

В таблице 2 приведён полный список возможных сетей при использовании 4-х байтового имени протокола IPv4. Первая строка таблицы (маска /32) выделяет только один сетевой узел. Вторая строка таблицы выделяет два адреса в сугубо специальном случае, определённом RFC 3021 – соединения точка-точка. И только последующие строки определяют сети с реальными адресами, реально используемые на практике.

**Таблица 2. Полный список возможных сетей IPv4 при использовании CIDR**

| IP/маска | Адресная часть IP-адреса | Маска | Всего адресов | Количество «реальных узловых» адресов | Класс |
|---|---|---|---|---|---|
| a.b.c.d/32 | +0.0.0.0 | 255.255.255.255 | 1 | (нет) | 1/256 C |
| a.b.c.d/31 | +0.0.0.1 | 255.255.255.254 | 2 | 2* | 1/128 C |
| a.b.c.d/30 | +0.0.0.3 | 255.255.255.252 | 4 | 2 | 1/64 C |
| a.b.c.d/29 | +0.0.0.7 | 255.255.255.248 | 8 | 6 | 1/32 C |
| a.b.c.d/28 | +0.0.0.15 | 255.255.255.240 | 16 | 14 | 1/16 C |
| a.b.c.d/27 | +0.0.0.31 | 255.255.255.224 | 32 | 30 | 1/8 C |
| a.b.c.d/26 | +0.0.0.63 | 255.255.255.192 | 64 | 62 | 1/4 C |
| a.b.c.d/25 | +0.0.0.127 | 255.255.255.128 | 128 | 126 | 1/2 C |
| a.b.c.0/24 | +0.0.0.255 | 255.255.255.000 | 256 | 254 | 1 C |
| a.b.c.0/23 | +0.0.1.255 | 255.255.254.000 | 512 | 510 | 2 C |
| a.b.c.0/22 | +0.0.3.255 | 255.255.252.000 | 1024 | 1022 | 4 C |
| a.b.c.0/21 | +0.0.7.255 | 255.255.248.000 | 2048 | 2046 | 8 C |
| a.b.c.0/20 | +0.0.15.255 | 255.255.240.000 | 4096 | 4094 | 16 C |
| a.b.c.0/19 | +0.0.31.255 | 255.255.224.000 | 8192 | 8190 | 32 C |
| a.b.c.0/18 | +0.0.63.255 | 255.255.192.000 | 16 384 | 16 382 | 64 C |
| a.b.c.0/17 | +0.0.127.255 | 255.255.128.000 | 32 768 | 32 766 | 128 C |
| a.b.0.0/16 | +0.0.255.255 | 255.255.000.000 | 65 536 | 65 534 | 256 C = = 1 B |
| a.b.0.0/15 | +0.1.255.255 | 255.254.000.000 | 131 072 | 131 070 | 2 B |
| a.b.0.0/14 | +0.3.255.255 | 255.252.000.000 | 262 144 | 262 142 | 4 B |
| a.b.0.0/13 | +0.7.255.255 | 255.248.000.000 | 524 288 | 524 286 | 8 B |
| a.b.0.0/12 | +0.15.255.255 | 255.240.000.000 | 1 048 576 | 1 048 574 | 16 B |
| a.b.0.0/11 | +0.31.255.255 | 255.224.000.000 | 2 097 152 | 2 097 150 | 32 B |

| IP/маска | Адресная часть IP-адреса | Маска | Всего адресов | Количество «реальных узловых» адресов | Класс |
|---|---|---|---|---|---|
| a.b.0.0/10 | +0.63.255.255 | 255.192.000.000 | 4 194 304 | 4 194 302 | 64 B |
| a.b.0.0/9 | +0.127.255.255 | 255.128.000.000 | 8 388 608 | 8 388 606 | 128 B |
| a.0.0.0/8 | +0.255.255.255 | 255.000.000.000 | 16 777 216 | 16 777 214 | 256 B = = 1 A |
| a.0.0.0/7 | +1.255.255.255 | 254.000.000.000 | 33 554 432 | 33 554 430 | 2 A |
| a.0.0.0/6 | +3.255.255.255 | 252.000.000.000 | 67 108 864 | 67 108 862 | 4 A |
| a.0.0.0/5 | +7.255.255.255 | 248.000.000.000 | 134 217 728 | 134 217 726 | 8 A |
| a.0.0.0/4 | +15.255.255.255 | 240.000.000.000 | 268 435 456 | 268 435 454 | 16 A |
| a.0.0.0/3 | +31.255.255.255 | 224.000.000.000 | 536 870 912 | 536 870 910 | 32 A |
| a.0.0.0/2 | +63.255.255.255 | 192.000.000.000 | 1 073 741 824 | 1 073 741 822 | 64 A |
| a.0.0.0/1 | +127.255.255.255 | 128.000.000.000 | 2 147 483 648 | 2 147 483 646 | 128 A |
| 0.0.0.0/0 | +255.255.255.255 | 000.000.000.000 | 4 294 967 296 | 4 294 967 294 | 256 A |

### 4.1.2. Приватные сетки или IP-адреса, не маршрутизируемые в Интернет (RFC 1918)

Приватные сетки или IP-адреса, не маршрутизируемые в Интернет (RFC 1918) — это адреса, которые кто угодно и когда угодно может использовать в своих сетях для своих целей. Они не уникальны в пределах всего мира, но они должны быть уникальны в пределах локальной / корпоративной сети. Да, корпоративной в том числе — см. выше пункт 2.3, ибо ничто не мешает построить корпоративную сеть, взяв за основу приватные сетки, ведь требования главного критерия, определяющего понятие корпоративной сети, не зависят от типа адресации в этой сети.

В таблице 3 перечислены приватные сетки общего назначения и их краткие описания.

Несмотря на рекомендацию RFC 6598, которая появилась не столь давно, очень часто провайдеры используют по старинке сеть 10.0.0.0.

Замечание. Формально, маршрутизаторы Интернета должны отбрасывать пакеты с адресами из этих сеток. Но на самом деле всё не совсем так и даже может оказаться совсем не так: всё определяется настройками этих маршрутизаторов и вполне можно (что иногда и делается) указать маршрутизатору пересылать подобные пакеты «куда надо».

**Таблица 3. Приватные сетки общего назначения**

| Сеть | Пояснение |
|---|---|
| **10.0.0.0/8** | Почти 17 млн. IP-адресов, которые можно использовать в своей локальной сети. Никаких особых рекомендаций для этой подсети нет. |
| **100.64.0.0/10** | Сеть на 4.194 млн. адресов, RFC 6598 рекомендует использовать эту сеть провайдерам, которые выпускают нас в Интернет. Если вы получаете от провайдера серый IP-адрес, то, скорее всего, он будет в этом диапазоне: от 100.64.0.1 до 100.127.255.254. |
| **172.16.0.0/12** | Немногим больше 1 млн. IP-адресов. Никаких особых рекомендаций для этой подсети нет. |
| **192.168.0.0/16** | Частная подсеть на 65 534 IP-адреса — или 256 сеток класса C. Никаких особых рекомендаций для этой подсети нет. |

### 4.1.3. Специальные приватные сетки и специальные IP-адреса

Специальные приватные сетки и специальные IP-адреса — это сетки и отдельные IP-адреса, для которых в спецификациях и стандартах прописано специальное назначение. Ниже все эти сетки и IP-адреса описаны в таблице 4. Те адреса, которых нет ни в этой, ни в предыдущей таблице (приватных сеток) являются публичными («белыми»).

**Таблица 4. Специальные сетки и адреса**

| **Сеть** | **Краткое описание** |
|---|---|
| **0.0.0.0/8** | Вы не можете нигде (за редким исключением) использовать IP-адреса, первый октет которых 0. Все эти адреса можно смело называть не маршрутизируемыми (от слова совсем) мета-адресами. Адреса из этой подсети используется для обозначения недопустимой, недостижимой цели (такой адрес вы можете увидеть в IP-пакете в поле IP-адрес назначения в том случае, когда узел сформировал пакет, но не знает куда конкретного его направить). Когда мы будем говорить о маршрутах и маршрутизации, мы увидим, что этот IP используется для задания маршрута по умолчанию. |
| **0.0.0.0/32** | Такую подсеть может использовать узел в качестве IP-адреса источника, когда делает запрос к DHCP-серверу. |
| **127.0.0.0/8** | Одна из самых бесполезных специальных IP подсетей, так как для задачи, которую она решает, было бы достаточно одного IP-адреса с маской /32. Любой адрес из этой сети является циклическим, иногда такой адрес называют локальным хостом или localhost. Если вам нужно реализовать схему взаимодействия клиент-сервер на одном компьютере в одной операционной системе без виртуализации, то, вероятно, вам придется использовать IP-адрес из этой подсети. |

| Сеть | Краткое описание |
|---|---|
| | Когда машина делает запрос к адресу из этой подсети, она делает запрос сама к себе, при этом физическая сетевая карта не используется. Если вам нужно проверить работу сетевых библиотек своего компьютера, то используйте адрес из данной подсети. Часто адрес из подсети 127.0.0.0/8 называют loopback-адресом, он есть, он всегда доступен, но ни один физический интерфейс за ним не закреплен. |
| **169.254.0.0/16** | Адреса из этого диапазона иногда называют канальными адресами, хотя это не совсем правильно переведено, оригинал раскрывает суть: Link-Local Address. В протоколе IPv6 с Link-Local Address придется работать часто, в IPv4 не очень. Сеть 169.254.0.0/16 используется в тех случаях, когда компьютер настроен на получение IP-адреса от DHCP-сервера, но по каким-то причинам не может его получить, в этом случае узел сам себе назначает IP из этой подсети, в надежде, что какой-нибудь узел из этой подсети тоже назначит себе такой адрес. Механизм работы узлов в такой ситуации описан в RFC 3927. |
| **192.0.0.0/24** | IETF просто взял и забрал у нас эту подсеть, в RFC 6890 об этом сказано |
| **192.0.0.0/29** | Dual-Stack Lite (DS-Lite). Описано в RFC 6333. Когда мы будем говорить про IPv6 разберемся с этой подсетью. |
| **192.0.0.170/32** | Используется для NAT из IPv6 в IPv4 и обратно. |
| **192.0.0.171/32** | DNS64, опять же тема из IPv6 и мы ее не касаемся сейчас. |
| **192.0.2.0/24 и 198.51.100.0/24** | Можно использовать только для примеров в документации. |
| **192.88.99.1/32** | Этот IP-адрес может использовать кто и угодно. На него узлы отправляют пакеты, когда хотят попасть из сети IPv4 в сеть IPv6. |
| **192.88.99.0/24** | В IPv6 очень хорошо реализован механизм распространения пакетов anycast, а вот эта сеть является костыликом IPv4, позволяющим эмитировать этот самый anycast. |
| **198.18.0.0/15** | Можно использовать только для тестов на производительность каналов связи и компьютерной сети. |
| **224.0.0.0/4** | Эта сеть используется для многоадресной рассылки (multicast), при этом стоит учесть, что глобально маршрутизируемыми являются подсети 233.0.0.0/8 и 234.0.0.0/8, а часть блоков из общей подсети являются зарезервированными, если интересно, то RFC 5771. |
| **240.0.0.0/4** | Это примерно 1/16 IP-адресов из всего пула IPv4, которые никому и никогда не давали и, наверное, не дадут, так как эти адреса зарезервированы для экспериментов. |
| **255.255.255.255/32** | Этот адрес часто используют узлы настроенные по DHCP, когда делают запрос к DHCP серверу, указывая его в поле IP-адрес назначения IP пакета. |

### 4.1.4. Автономные системы

Как сказано в пункте 3.2, адреса в «реальную экономику» выдают LIR (Локальные Интернет Регистраторы) — организации, называемые в обиходе провайдерами. Пример: РосНИИРОС, www.nic.ru. Но эта функция — перепродажа адресов дальше, не является обязательной. То есть, крупная организация вполне может получить в своё распоряжение некоторый блок адресов и распоряжаться им по своему усмотрению для своих внутренних целей.

LIR могут передавать желающим (в аренду) как отдельные адреса, так и целые сетки адресов из своего запаса. То есть, это розница.

А, вот, вышестоящие организации над ними — RIR, работают только оптом. То есть, предоставляют в аренду только сетки адресов, ранее минимальный блок адресов был 4096, сейчас, в связи с общей демократизацией, минимальный блок адресов — 512. Причём, RIR, предоставляя сетку адресов в аренду, одновременно присваивает этой сетке адресов Номер Автономной Системы (RFC 1930, RFC 4893), если такового номера ещё нет у данной организации, то есть, RIR определяет, что теперь данная организация, что приобрела (взяла в аренду) сетку адресов, одновременно обязуется ими управлять по определённым правилам (администрировать), о чём и заключается договор.

Номер Автономной Системы (ASN) — цифровой адрес Автономной системы, двухбайтовое число (short unsigned) в диапазоне от 0 до 65535 по старому RFC, или 4-байтовое число по новому RFC 4893. Некоторые адреса из этого диапазона зарезервированы: 0, 65535 — зарезервированы IANA, 64496-64511 — для использования в документации и примерах, 64512-65534 — приватные номера Автономных Систем.

Таким образом, каждый провайдер (LIR) и вообще каждая организация, арендующая таким образом блок адресов, одновременно являются администраторами Автономной Системы.

Стать LIR могут не только лишь все, не каждый может им стать, поскольку для этого необходимо иметь достаточно ресурсов, прежде всего финансовых.

Пример — см. рис. 30. Прежде всего придётся оплатить годовую аренду как минимум 512 адресов, а это как минимум несколько миллионов рублей при текущем курсе. Но чтобы ими воспользоваться (хотя бы для выхода в Интернет), необходимо их некоторым образом подключить к Интернет. То есть, нужно подключить к Интернет некоторую Автономную Систему. Для этого нужно приобрести достаточно недешёвый маршрутиза-

тор, поддерживающий, например, протокол BGP (протокол пограничной маршрутизации), арендовать канал связи и заключить договор как минимум с одним LIR о взаимообмене трафика. Трафик, как правило, оплачивается по «разностной» схеме: за входящий платите вы, за исходящий оплачивает противоположная сторона, к расчёту — разница трафика.

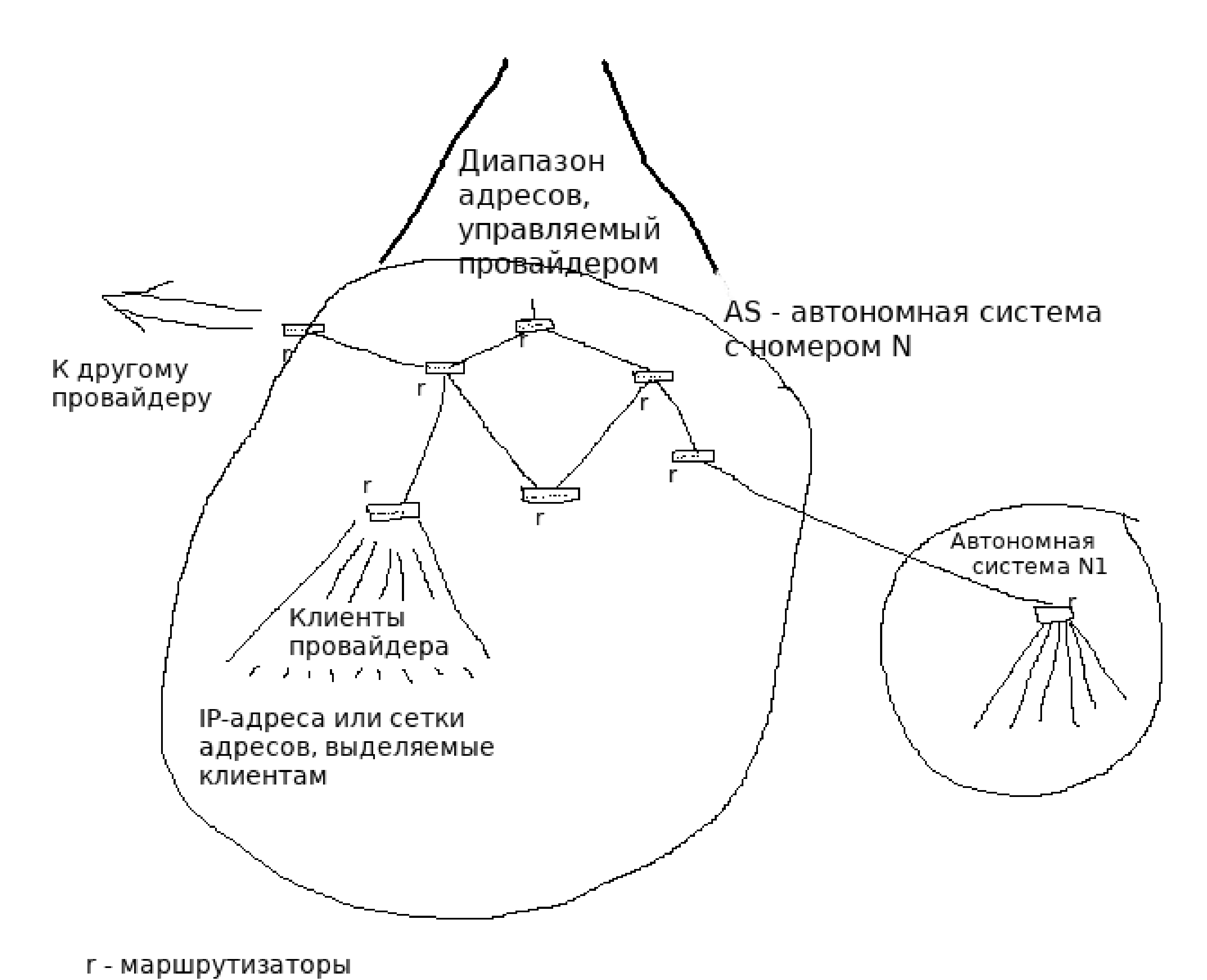



*Рис. 30.* Провайдеры и Автономные Системы

**Вывод.** С точки зрения цифрового именования весь Интернет поделен на множество (в настоящее время — несколько тысяч) Автономных Систем, взаимодействующих друг с другом через пограничные маршрутизаторы. То есть, зоной ответственности провайдеров являются их Автономные Системы, которые они администрируют. Иначе говоря, предметом труда провайдера с точки зрения цифрового именования является его Автономная Система, включая пограничные маршрутизаторы.

Как правило, провайдеры (они же LIR) администрируют одну Автономную Систему, хотя бывают исключения, если провайдер достаточно долго на рынке и владеет достаточно большим объёмом адресов.

## 4.2. Функции провайдеров

### 4.2.1. Что делает провайдер Интернета с точки зрения цифрового именования

Провайдер Интернета реализует следующие функции:

- выделение клиенту цифрового имени или группы имён (сетки адресов) «на период времени» — сдача в аренду,
- предоставление возможности клиенту доступа к другим именам в Интернет — выход в пространство цифровых имён Интернет,
- обеспечение возможности видимости цифрового имени клиента (возможно, провайдером же сформированного, а возможно полученного клиентом другим путём) в пространстве цифровых имён Интернет,
- обеспечение возможности доступа из Интернет к хосту клиента (точнее, к запущенному на хосте сервису) по цифровому имени,
- и др.

## 4.3. Кто «рулит» в Интернет

### 4.3.1. Организации системы управления доменными именами и IP-адресами

1. Во главе всей этой схемы стоит организация, называемая **ICANN (Internet Corporation for Assigned Names and Numbers)** или корпорация по управлению доменными именами и IP-адресами, говорят, что она некоммерческая, но создана при участии правительства США, как понятно из названия, для поддержания порядка в хаосе Интернет-адресов. Из названия также понятно, что у нее два основных направления, в которых можно вести некоммерческую деятельность. До осени 2016 года у ICANN был контракт с Министерством торговли США и Национальным управлением информации и связи США, то есть, до 2016 года правительство США могло контролировать Интернет. Сейчас контракт истёк и не продлён. Но фактически американская компания Verisign по-прежнему обеспечивает поддержку серверов DNS и ведёт реестр доменов, хотя формально администрирование интернета в ведении ICANN.

2. **IANA (администрация цифрового адресного пространства Интернета)**, эта структура подчиняется непосредственно ICANN и занимает-

ся контролем IP-адресов во всем мире. В ее задачи входит распределять IP-адреса и номера AS между регионами планеты Земля. Но распределяет она не абы кому, а только конкретным организациям, которые являются главными в своем регионе, их называют RIR, например.

3. **RIR (региональная регистратура Интернета)**. У RIR можно запросить только очень крупный блок IP-адресов, также RIR выдает номера автономных систем и оформляет так называемых LIR. Всего в мире пять RIR (см. рис. 35): **AfriNIC (Африка), RIPE NCC (Европа, Россия и Ближний восток), APNIC (Австралия и вся остальная Азия), ARIN (США и Канада), LACNIC (Южная Америка, Центральная Америка и Мексика)**. Кто угодно RIRом стать не может, его назначает IANA, RIRы не продают IP-адреса, они лишь регистрируют LIRы и решают конфликтные ситуации, если кто-то где-то накосячит, но LIRы платят своими RIRам членские взносы.

4. **LIR или локальный регистратор**, к которому можно «подкатить на хромой кобыле» и купить несколько сотен PI-адресов (PI-адреса — провайдеро-независимые адреса (Provider Independed), то есть, полученные непосредственно у LIR). LIRом может стать любой желающий, платите только деньги, раньше чтобы стать LIR, нужно было взять 4096 IP-адресов. LIRы делятся на пять категорий, в зависимости от количества IP-адресов, которые у них есть: Extra Large, Large, Medium, Small и Extra Small. Категорию присваивает RIR.

5. **NIR есть только в некоторых странах**, например, в Китае, N – национальный.

Стоит сказать, что если вы не провайдер или не крупный дата-центр, то за PI-адресами вам придется бежать в LIR или самому становится LIR, но если вы не провайдер и не дата-центр, то, вероятно, вам это и не нужно.

6. **И ещё нужно сказать про DNSSEC** (Domain Name System Security Extensions — Систему безопасного расширения доменного имени), созданную в июле 2010. Эта Система не только увеличивает сохранность пользовательской информации в Сети, но позволяет отключить Интернет в случае чрезвычайного происшествия — катастрофы или теракта на телекоммуникационном объекте. Для этого создана группа из семи «избранных» специалистов, которые могут воспользоваться специальными картами, которые являются частичками единого ключа, и вместе перезапустить Мировую сеть. Известен один из них — Пол Кейн – преподаватель в университете английского города Бас. Пока неизвестно, кто получил шесть оставшихся фрагментов ключа к рубильнику Интернета. Смотри также пункт 5.2.2 и рис. 31.

**Вопросы «на засыпку»**

1. 127.0.0.1 — это сетка или адрес?

2. В УлГУ-шной сети (на Свияге) для gw используется адрес 10.2.0.1/16. Какая сетка адресов при этом используется?

3. Как называется организация, предоставляющая услуги присоединения пользователей к сети Internet.

4. Можно ли «выключить» Интернет? — см. рис. 31-1 ниже.

5. Поскольку все адреса в интернете объединены в глобальную сеть, то система именования должна обладать свойством . . . (каким?).

6. На следующем рисунке 31 покажите пальчиком ошибку:

**Протокол TCP/IP**

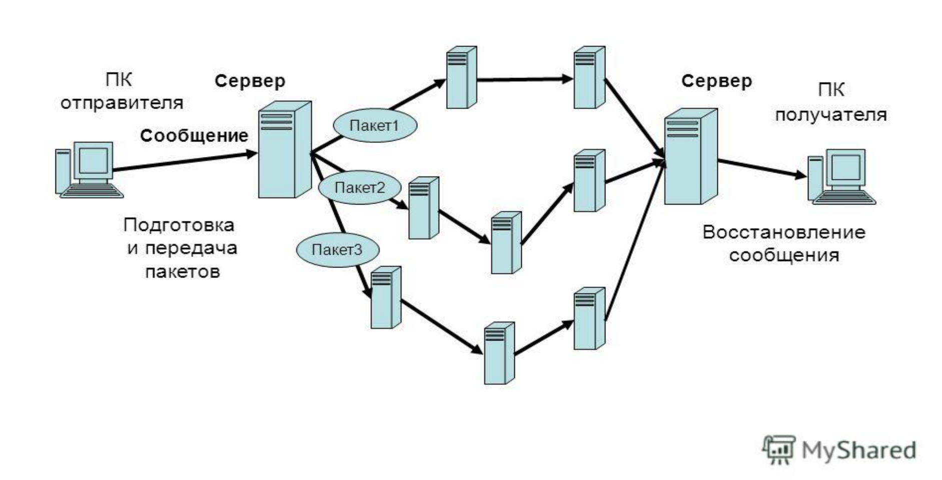



*Рис. 31.* Где ошибка?




*Рис. 31-1.* «Ключ» от Интернет или одна седьмая «выключателя Интернета»

# Лекция 5. Стек TCP/IP. Символьная форма имени

## 5.1. Именование взаимодействующих объектов в стеке TCP/IP

### 5.1.1. Требования к именованию сетевых объектов

Таким образом, важнейшей задачей стека TCP/IP является создание глобальной сети посредством объединения в единое целое множества отдельных локальных сетей, а иногда и не только локальных.

Задача сложная, для её реализации используются различные технические решения аппаратного и программного уровня. И среди этих решений одним из важнейших является разработка и реализация принципов именования взаимодействующих сетевых объектов в этой большой глобальной сети. «Как корабль назовёшь, так он и поплывёт». То есть, именование объектов должно быть эффективным:

- с одной стороны, оно должно легко и качественно реализовываться в аппаратно-программном сетевом обеспечении и, соответственно, быть лёгким в сопровождении, что особенно важно, поскольку касается глобальной сети, в которой сетевых узлов больше чем дохера;

- с другой стороны, оно должно быть удобным пользователям этой глобальной сети, ведь, в конечном итоге, делается всё для человека.

Поэтому, к адресу сетевого узла и схеме его назначения предъявляются определённые требования [27}:

1) адрес должен уникально идентифицировать компьютер в сети любого масштаба;

2) схема назначения адресов должна сводить к минимуму ручной труд администратора и вероятность дублирования адресов;

3) адрес должен иметь иерархическую структуру, удобную для построения больших сетей; в больших сетях, состоящих из многих тысяч узлов, отсутствие иерархии адреса может привести к большим издержкам — конечным узлам и коммуникационному оборудованию придется оперировать с таблицами адресов, состоящими из тысяч записей;

4) адрес должен быть удобен для пользователей сети, а это значит, что он должен иметь символьное представление например, comp12.lab326.ulsu.ru;

5) адрес должен иметь по возможности компактное представление, чтобы не перегружать память коммуникационной аппаратуры — сетевых адаптеров, маршрутизаторов и т. п.

### 5.1.2. Формы имени объекта

На канальном уровне (или в терминах стека TCP/IP — на уровне сетевых интерфейсов) сетевые узлы именуются MAC-адресами. Однако, поскольку канальный уровень — это уровень сетевых технологий (локальных сетей), то видимость этих имён сетевых узлов — локальная (в силу ограничений, рассмотренных в лекции 1). И поскольку глобальная сеть (создаваемая стеком TCP/IP) — это сеть локальных сетей, то получается, что в каждой отдельной локальной сети именование есть, но как обратиться с некоторого компа в некоторой локальной сети к некоторому компу в некоторой другой локальной сети — неясно.

Эту задачу как раз и решает сетевой уровень стека TCP/IP, конкретно, решение этой задачи возложено на протокол IP.

Протокол IP по своим правилам присваивает сетевому узлу имя. Правила формулируют администраторы сети и либо вводят вручную через интерфейс сетевого программного обеспечения, либо фиксируют в конфигурационных файлах этого сетевого ПО (требование 2; см. пункт 2.4.1). Это имя строго уникально (требование 1; см. пункт 2.4.1) и идентифицирует узел в создаваемой стеком TCP/IP глобальной сети.

Но формат имени определяет реализация протокола IP. Например, в IPv4 (протоколе IP версии 4) формат имени сетевого узла следующий:

а) имя сетевого узла имеет две формы: символьную и цифровую;

б) обе формы имени равнозначны, но используются для разных целей;

в) символьная форма имени предназначена для человека и удовлетворяет требованиям 3 и 4 (см. пункт 2.4.1);

г) цифровая форма имени предназначена для сетевого программного обеспечения и удовлетворяет требованиям 3 и 5.

**Важно!** Основной является цифровая форма имени — именно с ней и только с ней работает протокол IP (требование 1, 3 и 5; см. пункт 2.4.1). Исторически эта форма имени появилась первой вместе с протоколом. Пересылку данных и их маршрутизацию в Интернет осуществляет протокол IP и ему для решения этой задачи никакого другого именования не надо.

**Важно!** Символьная форма имени является дополнительной, появилась чуть позже вместе с появлением DNS. То есть, DNS работает с обеими формами имени, в сущности DNS и был создан по причине появления вто-

рой, символьной формы имени и обеспечения (автоматизации) процесса преобразования форм имён («разрешения» имён). Причина появления этой второй формы имени сетевых узлов — специфика мышления человека: мы, люди, очень хорошо запоминаем (обрабатываем) символьные имена и очень плохо цифровые (требование 1, 3, 4 и 5; см. пункт 2.4.1).

### 5.1.3. Ограничения символьной формы имени

Компоненты имени — имя хоста (hostname) или имя домена. В полном доменном имени они разделяются точками.

Пример: comp1.lab326.ulsu.ru — здесь 4 компонента имени, из которых первое — hostname. Такое имя компа также называется каноническим именем компа. Формально (в соответствии с RFC) все четыре компонента имени называются доменами или доменными именами. Однако в обиходе мы всё-таки выделяем первое доменное имя и даже придумали ему отдельное названия — hostname. Это потому, что оно часто используется отдельно от всего остального, самостоятельно, и в обиходе, и в конфигурационных файлах. Полное (каноническое) доменное имя может заканчиваться точкой: comp1.lab326.ulsu.ru., тогда оно называется FQDN — полностью квалифицированное доменное имя (см. рис. 32).

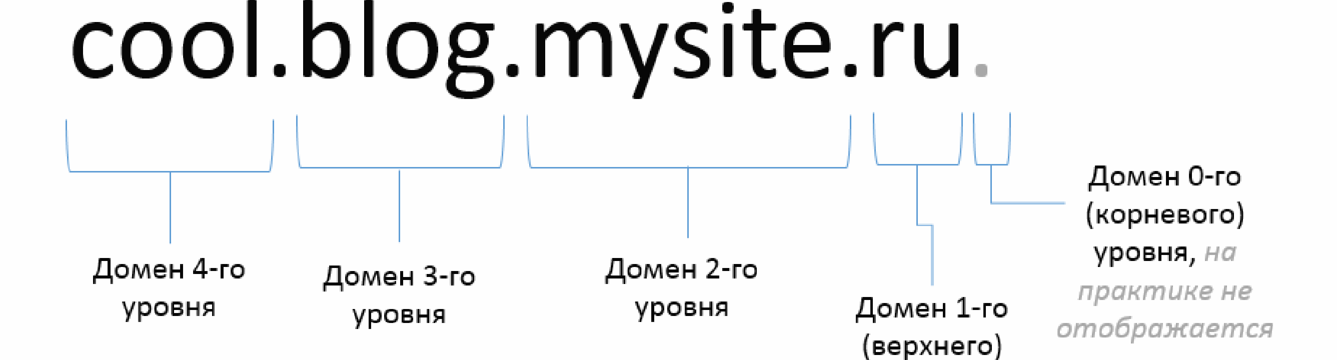



*Рис. 32.* Структура символьного имени

В соответствии с RFC 952 и RFC 1123 символьное имя сетевого узла — строка символов с алфавитом: [a..z0..9-.]. То есть, буквы в нижнем регистре, цифры, тире и точка. Причём, точка имеет особый смысл — это разделитель компонент имени. В символьном имени не могут использоваться следующие запрещённые символы: запятая (,), тильда (~), двоеточие (:), восклицательный знак (!), знак «собачка» "@", знак номера (#), знак доллара ($), процент (%), символ "крышка" (^), амперсанд (&), апостроф

('), точка (.), круглые скобки (()), фигурные скобки ( {} ), подчеркивание (_), пустое пространство (пробел).

RFC 2181 «Разъяснения к спецификации DNS» расширяет набор символов, разрешенный в именах DNS. В нем указано, что компонент имени (метка DNS) может быть любой двоичной строкой и ее не обязательно интерпретировать как набор символов ASCII. В документах RFC 2671 и RFC 2373 можно посмотреть дополнительную информацию.

Первый символ компоненты имени должен быть алфавитным или числовым. Последним символом не должен быть знак тире или точка. Минимальная длина компонента: 2 символа. Максимальная длина компонента: 63 символа. Длина полного доменного имени составляет 63 байта на компонент и 255 байтов для полного доменного имени. Таким образом, максимальная вложенность поддоменов равна 255 / 3 = 84. Конечно, такую вложенность никто не использует, обычно ограничиваются 3, 4 или 5 уровнями. На рис. 33 показан пример использования имени 6-го уровня. Почему не используют большую вложенность? Ну, так это же ручками вводить надо. Поэтому, чем короче имя — тем оно ценнее.

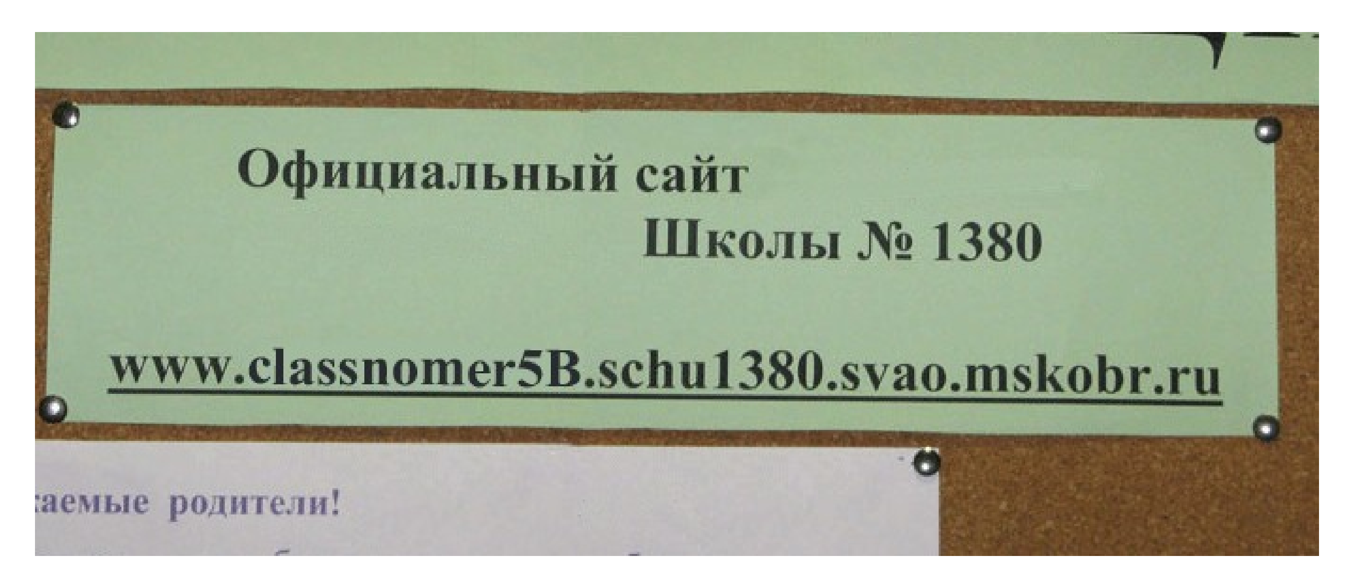



*Рис. 33.* Имя сетевого узла шестого уровня вложенности

Существуют зарезервированные слова и буквосочетания, случайное использование которых может привести к ошибкам.

Именование сетевых узлов имеет непосредственное отношение к реализации DNS. За полувековую историю стека TCP/IP (и, соответственно, Интернета) релизов DNS было несколько и сделаны они были разными организациями/фирмами. Это имеет значение.

### 5.1.4. Resolver, назначение и его работа

resolver — набор функций (около десятка) из системной библиотеки (glibc), которые позволяют прикладной программе, скомпонованной с ними, получать по доменному имени IP-адрес компьютера или по IP-адресу доменное имя. То есть, выполнять преобразование из одной формы имени в другую. Сами эти функции обращаются к системной компоненте resolver, которая либо обрабатывает свои конфигурационные файлы, либо ведет диалог с сервером доменных имен и таким образом обслуживает запросы прикладных программ пользователя.

resolver имеет три конфигурационных файла: host.conf, hosts, resolv.conf, расположенных в каталоге /etc. В последних версиях (2 и выше) вместо файла host.conf рекомендуется использовать файл nsswitch.conf.

**Файл host.conf** определяет общий порядок (последовательность опроса баз) разрешения имён. Например, он может выглядеть так:

```
order hosts,bind
multi on
```

В данном случае говорится, что сначала resolver заглянет в локальную базу имён — файл /etc/hosts, а потом, в случае неуспеха (!), он обратится к bind, то есть к DNS. Могут быть указаны и другие базы данных, но это более сложные случаи. Вторая строчка определяет, что данный комп может иметь несколько имён — именование в стеке TCP/IP подобную вольность позволяет.

**Файл hosts** — это локальная база данных, в которой администратор описывает свою сеть. Файл может содержать, например, следующее:

```
127.0.0.1          localhost.localdomain  localhost
192.168.199.111    pay.acca.firma.ru      pay
192.168.199.112    keeper.acca.firma.ru   keeper
192.168.199.113    general.acca.firma.ru  general
192.168.199.114    accer1.acca.firma.ru   accer1
192.168.199.115    accer2.acca.firma.ru   accer2
192.168.199.116    i.tak.dalee.ru         i
```

Здесь первая строчка — определение адреса и имени для интерфейса локальной петли (lo0). **Эта строчка должна быть всегда именно такой и никакой другой.**

Последующие строчки — определение адресов и имён для интерфейсов сетевых плат всех компьютеров сети (интерфейсов eth0, в ALTLinux этот интерфейс называется enp3s0) в предположении, что для локальной сети выбрана сетка 192.168.199.0.

*Замечание 1.* Файл hosts должен содержать определение интерфейсов для **всех** компьютеров локальной сети, если какой-либо компьютер не будет описан в этом файле или в строчке описания будет ошибка, то этот компьютер не будет виден в сети и, следовательно, будет недоступен.

*Замечание 2.* Файл hosts должен быть **одинаковым** на всех компьютерах сети, в том числе, на тех, на которых установлена ОС Windows. Точнее сказать, на всех компах, на которых установлен стек TCP/IP и которые должны быть подключены к данной сети. Например, в ОС Windows XP этот файл находится по пути C:\windows\system32\drivers\etc\ и по умолчанию называется hosts.SAM (SAM, от sample — пример). Его необходимо исправить, как указано выше, и сохранить под именем hosts.

*Замечание 3.* В конце файла должен стоять символ перевода на новую строку, иначе могут быть проблемы с последним адресом.

*Замечание 4.* Разделителем между полями в файле hosts является символ Tab. Внимание: не пробел, а Tab! Иначе возможны проблемы — некоторые релизы библиотеки resolver`а могут не понимать пробелы в этом конфигурационном файле.

**Файл resolv.conf** используется в том случае, если resolver не смог выполнить преобразование имён с помощью локальной базы имён (файла hosts) и вынужден перейти к следующему шагу — обратиться к bind. Файл resolv.conf может содержать, например, следующее:

```
search example.com
nameserver 192.168.0.1
nameserver 10.2.0.1
```

В данном случае говорится, что доменом поиска является exanple.com. То есть, наш комп находится в домене example.com и мы можем обращаться к другим компам нашего поддомена просто по hostname, не указывая домена, доменная часть имени будет добавлена автоматически.

В следующих строках указаны адреса DNS серверов, к которым resolver будет обращаться с запросами на преобразование имен, сначала к первому, а если он не ответит в течение 5 секунд, то ко второму.

## 5.2. Понятия домена и зоны

### 5.2.1. Домены и зоны

Домен — это все множество машин, которые относятся к одному и тому же доменному имени. Например, все машины, которые в своем имени имеют постфикс ulsu.ru относятся к домену ulsu.ru. Однако, сам домен разбивается на поддомены или, как их еще называют, зоны. У каждой зоны может быть свой собственный сервер доменных имен, сервер DNS. Разбиение домена на зоны и организация сервера для каждой из зон называется делегированием прав управления зоной соответствующему серверу доменных имен, или просто делегированием зоны (см. рис. 23).

Вообще говоря, понятия "зона" и "домен" носят локальный характер и отражают только тот факт, что при настройках и взаимодействии между собой серверы доменных имен исходят из двухуровневой иерархии. Это значит, что если в зоне необходимо создать разделение на группы, то зону можно назвать доменом, и в нем организовать новые зоны. Например, у компании Sun Microsystems есть зона russia в домене sun.com (russia.sun.com). Так вот, эту зону тоже можно разбить на зоны, например, info.russia.sun.com и market.russia.sun.com.

**Пример 1.** Пусть некоторая небольшая фирма зарегистрировала имя (домен) 2-го уровня firma.su для своей локальной (пока ещё) сети. В определённой выше терминологии это имя firma.su является доменным именем 2-го уровня, или просто доменом 2-го уровня. Он же является зоной firma.su. Предположим также, что в этой фирме работает некий админ Вася, который по поручению и собственному наитию установил и настроил в фирме сервер DNS с именем ns1.firma.su, расположив его непосредственно в этом домене. В конфигурационных файлах этого сервера он описал все компы фирмы, например так:

| | |
|---|---|
| ns1.firma.su | - сервер DNS фирмы; |
| gw.firma.su | - выход в Инет; |
| admin.firma.su | - рабочее место админа Васи; |
| boss.firma.su | - комп хозяина; |
| realdir.firma.su | - комп исполнительного директора; |
| genacca.firma.su | - комп главбуха; |
| pay.firma.su | - комп бухгалтера по расчёте зарплаты; |
| keeper.firma.su | - комп кладовщика; |
| disp.firma.su | - комп диспетчера в производственном отделе (в цехе); |
| agent.firma.su | - комп агента в отделе снабжения; |
| itakdalee.firma.su | - и так далее. |

Следовательно, админ Вася сопровождает домен фирмы firma.su, который состоит из одной зоны firma.su и которые олицетворяют локальную сеть этой фирмы. Конец примера 1.

**Пример 2.** Предположим далее, что за год фирма выросла, в ней явно оформились подразделения (отделы и цеха), оборот превысил сотню млн, в результате чего хозяина охватила паранойя на почве (пока ещё только) финансовой безопасности, да и Вася качественно вырос и поимел нужную квалификацию и потому по общему разумению локальная сеть фирмы была преобразована в корпоративную. Вследствие этого Вася внёс изменения:

- добавил ещё один сервер DNS ns2.firma.su;
- завёл поддомены для подразделений, для каждого свой поддомен, типа так:

| | |
|---|---|
| owner.firma.su | - руководство; |
| adm.firma.su | - поддомен Васи; |
| acca.firma.su | - бухгалтерия; |
| market.firma.su | - отдел маркетинга; |
| sale.firma.su | - отдел продаж; |
| factory.firma.su | - производственный отдел; |
| store.firma.su | - склад; |
| logis.firma.su | - отдел снабжения. |

В соответствии с новой структурой фирмы Вася внёс изменения в конфигурационные файлы сервера DNS примерно так:

| | |
|---|---|
| ns1.firma.su | - 1-ый сервер DNS фирмы; |
| ns2.adm.firma.su | - 2-ой сервер DNS фирмы; |
| gw.firma.su | - выход в Инет; |
| www.firma.su | - внутренний веб-сервер фирмы; |
| admin.adm.firma.su | - рабочее место админа Васи; |
| boss.owner.firma.su | - комп хозяина; |
| realdir.owner.firma.su | - комп исполнительного директора; |
| genacca.acca.firma.su | - комп главбуха; |
| pay.acca.firma.su | - комп бухгалтера по расчёте зарплаты; |
| keeper.store.firma.su | - комп кладовщика; |
| disp.factory.firma.su | - комп диспетчера в производственном отделе (в цехе); |
| agent1.logis.firma.su | - комп агента в отделе снабжения/продаж (логистики); |
| i.tak.dalee.firma.su | - и так далее. |

Следовательно, админ Вася сопровождает домен фирмы firma.su, разбитый на поддомены. На серверах DNS в конфигурационных файлах описаны зона firma.su и зоны owner.firma.su, adm.firma.su, acca.firma.su, market.firma.su, factory.firma.su, store.firma.su, logis.firma.su, dalee.firma.su и которые олицетворяют теперь уже корпоративную сеть этой фирмы. Конец примера 2.

**Пример 3.** Предположим далее, что ещё через год фирма снова выросла, занимает уже несколько зданий, компов в ней уже больше сотни, оборот о-го-го скока млн, в результате чего хозяин озадачен не только паранойей на почве финансовой безопасности, но и безопасности бизнеса вообще, да и Вася превратился в существенно квалифицированного специалиста и потому уже не справляется с поддержкой ИТ-шного зоопарка фирмы и поэтому по общему разумению в фирме появились ещё пара/тройка админов в подразделениях. Админы подразделений создали свои серверы DNS, обслуживающие свои подразделения. Вследствие этого Вася внёс изменения в конфиги своих серверов DNS:

- делегировал соответствующие зоны на серверы DNS подразделений, оставив в своём ведении зоны только тех подразделений, где своих админов ещё не было.

Следовательно, админ Вася теперь сопровождает домен фирмы firma.su, разбитый на поддомены. На серверах DNS, сопровождаемых Васей, в конфигах описаны зона firma.su и некоторые зоны подразделений. По делегированным зонам в конфигах Васиных серверов определены ссылки на соответствующие сервера DNS подразделений, сопровождающие эти зоны. Соответственно, админы подразделений, сопровождающие свои сервера DNS, теперь сопровождают свои поддомены с соответствующими зонами. В конфигах этих серверов DNS определены ссылки (forward) на главные сервера DNS фирмы, сопровождаемые Васей. То есть, корпоративная сеть этой фирмы приобрела новый статус. Конец примера 3.

Таким образом, понятие домена почти равно понятию зоны. Отличия в том, что, когда говорят про домены — говорят о символьном именовании сетевых узлов в стеке TCP/IP вообще, а когда говорят про зоны — определённо говорят не просто про именование, а конкретно как работает DNS с именами в стеке TCP/IP.

### 5.2.2. Корневая зона

Корневая зона Интернет обозначается именем «.» — точка (см. рис. 36 ниже в пункте 5.2.4.).

**Корневые серверы DNS** — DNS-серверы, обеспечивающие работу корневой зоны DNS в сети Интернет. Корневые серверы DNS отвечают на запросы других DNS-серверов в ходе преобразования доменных имён в IP-адреса и позволяют получить список DNS-серверов для любого домена верхнего уровня (TLD): RU, SU, COM, NET, MUSEUM, INFO и др.

Существует тринадцать корневых серверов DNS, их доменные имена имеют вид *letter*.root-servers.net, где *letter* — буква от a до m (см. информационные сайты http://*letter*.root-servers.org). Каждый корневой сервер DNS состоит из множества хостов-реплик (зеркал), размещаемых в различных местах сети Интернет и имеющих один IP-адрес (см. рис. 34). По состоянию на 19.09.20 количество зеркал — 1326. Маршрутизация запросов к репликам корневых серверов DNS осуществляется с применением технологии anycast (RFC3258). Таким образом достигается быстрое время отклика и стабильная работа системы.

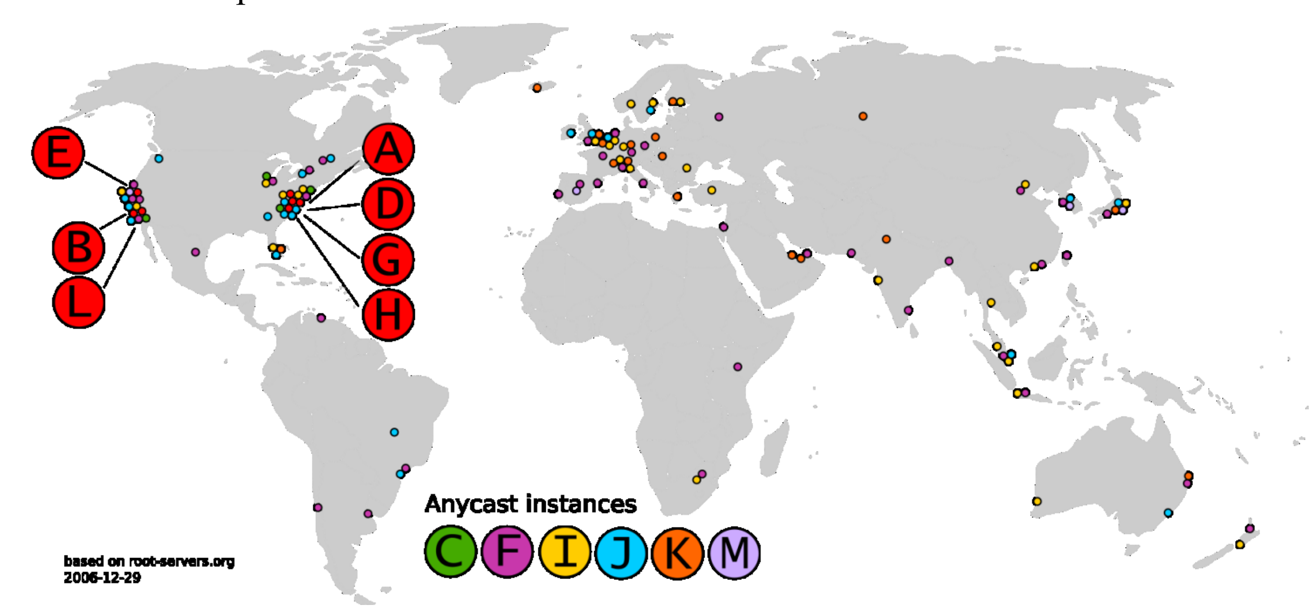



*Рис. 34.* Расположение корневых серверов DNS. Однако, данная карта показывает состояние на 29.12.2006

Почему корневых серверов DNS именно 13? А потому что, максимальный размер пакета протокола DNS — 512 байт. И в этот размер могут поместиться только 13 имён корневых серверов. Поэтому, пока протокол DNS таков, соответственно и корневых серверов будет только 13.

Корневые серверы DNS управляются двенадцатью различными организациями, действующими на основании соглашений с корпорацией

ICANN. В их число входят университеты, организации Министерства Обороны США, некоммерческие ассоциации. Операторы корневых серверов DNS (формально) финансово и юридически независимы от ICANN и образуют неформальную группу, целью которой является координация совместных действий и обмен операционной информацией и опытом. Члены группы являются также членами Консультационного совета ICANN по управлению корневыми серверами (Root Server System Advisory Committee, RSSAC), в задачу которого входит выработка рекомендаций по управлению корневыми серверами DNS и внесению различных изменений в систему. Принято считать, что подобная независимость и разнородность операторов корневых серверов DNS является основой технической и политической стабильности системы в целом, исключая узурпацию управления какой-либо из сторон. Обратите внимание на формулировку: «принято считать». То есть, «считать принято», но гарантий от злоупотреблений нет. Тем более, учитывая, кто является управляющими организациями (см. таблицу 5).

Официальная информация о действующих корневых серверах DNS публикуется на сайте Ассоциации операторов Корневых серверов DNS http://root-servers.org.

**Таблица 5. Корневые серверы DNS**

| Имя хоста | IP-адреса (IPv4, IPv6) | Кол-во узлов сервера | Управляющая организация | Местоположение упр. организ. |
|---|---|---|---|---|
| a.root-servers.net | 198.41.0.4, 2001:503:ba3e::2:30 | 16 | VeriSign, Inc. | Dulles, Virginia, США |
| b.root-servers.net | 199.9.14.201, 2001:500:200::b | 6 | University of Southern California (ISI) | Marina Del Rey, California, США |
| c.root-servers.net | 192.33.4.12, 2001:500:2::c | 10 | Cogent Communications | Вашингтон, США |
| d.root-servers.net | 199.7.91.13, 2001:500:2d::d | 149 | University of Maryland | Колледж-Парк, Мэриленд, США |
| e.root-servers.net | 192.203.230.10, 2001:500:a8::e | 254 | NASA (Ames Research Center) | Маунтин-Вью, Калифорния, США |
| f.root-servers.net | 192.5.5.241, 2001:500:2f::f | 242 | Internet Systems Consortium, Inc. | University of Southern California, Калифорния, США |
| g.root-servers.net | 192.112.36.4, 2001:500:12::d0d | 6 | US Department of Defense (NIC) | Колумбус, Огайо, США |

| Имя хоста | IP-адреса (IPv4, IPv6) | Кол-во узлов сервера | Управляющая организация | Местоположение упр. организ. |
|---|---|---|---|---|
| h.root-servers.net | 198.97.190.53, 2001:500:1::53 | 8 | US Army (Research Lab) | Aberdeen Proving Ground, Maryland, США |
| i.root-servers.net | 192.36.148.17, 2001:7fe::53 | 64 | Netnod | Netnod Internet Exchange i Sverige, Швеция |
| j.root-servers.net | 192.58.128.30, 2001:503:c27::2:30 | 118 | VeriSign, Inc. | Dulles, Virginia, США |
| k.root-servers.net | 193.0.14.129, 2001:7fd::1 | 73 | RIPE NCC | европейский региональный регистратор Интернет, Амстердам, Нидерланды |
| l.root-servers.net | 199.7.83.42, 2001:500:9f::42 | 147 | ICANN | Лос-Анджелес, Калифорния, США |
| m.root-servers.net | 202.12.27.33, 2001:dc3::35 | 5 | WIDE Project | Keio University, Tokyo, Япония |

На 22.05.2018 в России размещено 11 реплик (зеркал) корневых серверов DNS, в том числе:

f.root (Москва — 2 шт.);

i.root (Санкт-Петербург);

j.root (Москва, Санкт-Петербург);

k.root (Москва, Санкт-Петербург, Новосибирск);

l.root (Москва, Ростов-на-Дону, Екатеринбург).

Функционирование корневых серверов DNS критично для функционирования сети Интернет в целом, поскольку они обеспечивают первый шаг в трансляции доменных имён в IP-адреса и потому используется достаточно мощная аппаратная база, например, такая:

- коммутирующее ядро (Routers Cisco 72XX; Switches Cisco Catalyst 35XX)
- серверный кластер (Сервера Intel Xeon, 3 GHz, RAID)
- сервер статистики (Сервера Intel Xeon, 3 GHz, RAID).

В качестве программного обеспечения почти на всех серверах используется пакет BIND, кроме H, K, L, на которых установлен пакет NSD.

Опровержение распространённых заблуждений:

- за исключением незначительной доли DNS-запросов, интернет-трафик не проходит через корневые сервера;

- не каждый DNS-запрос обрабатывается корневым сервером;
- корневые серверы обслуживаются не добровольцами в качестве хобби, а профессионалами, и хорошо финансируются; пожалуй, админы этих серверов — это самые высокооплачиваемые ИТ-шники, зарплаты до миллиона руб. в месяц (в месяц!);
- ни одна организация (коммерческая или правительственная) не контролирует всю систему (но говорят, что это не точно).

По разным оценкам, только от 18 до 32 % разрешений доменных имён приводит к обращению непосредственно к одному из корневых серверов, остальные запросы используют кэшированные DNS-записи о TLD NS на нижестоящих серверах.




*Рис. 34-0.* Выключить Интернет? Это очень просто!

Таким образом, технически корневая зона глобальной DNS обслуживается децентрализованной системой из множества физических серверов, расположенных в различных странах мира и физически управляемых различными организациями. Такая система сложилась исторически в результате заключения соглашений между многими заинтересованными сторонами. Очевидно, тут есть масса преимуществ: децентрализованная (в плане компьютерного обеспечения), распределенная по земному шару система надежнее и гибче, она масштабируется с меньшими сложностями, она лучше реагирует на изменения нагрузки. Кроме того, считается, что децентрализованный подход делает Интернет более демократичным (да-да, и здесь это!).

Однако не стоит спешить с выводами о том, что корневая зона DNS — это децентрализованная система, не имеющая центрального управления. На первый взгляд такое замечание кажется парадоксальным: ведь корневых серверов множество и они не только принадлежат разным компаниями, но и находятся в разных странах мира. Вся хитрость в том, что физические компьютеры и данные, которые на этих компьютерах находятся. — это две большие разницы.

Для сохранения целостности системы имен корневые серверы должны содержать одинаковую информацию о корневой зоне. Это очень важный момент, политическая составляющая которого никак не меньше технической. Поэтому вся распределенная система корневых серверов DNS получает сведения о корневой зоне из единственного источника. Этим источником является специальный защищенный скрытый сервер, находящийся под управлением компании из США VeriSign (см. пункт 4.3) и физически располагающийся на территории военной базы в США. Все корневые серверы по расписанию копируют сведения об адресации со скрытого сервера-источника и без изменений (это строго оговорено) передают данную информацию ее потребителям в Интернете. Поэтому, после отключения этого скрытого сервера Интернет должен выключиться через некоторое время (обычно, несколько часов). Везде. Возможно, кроме Китая. Авторы не располагают сведениями о том, создан ли «скрытый» сервер в РФ. И именно поэтому «атаки на Интернет» (DDoS- атаки, о которых вы, вероятно, слышали) не могут привести к выключению Интернет вообще, а максимум к временной недоступности Интернет в отдельных странах, что и было продемонстрировано в недалёком прошлом.

Отключением отдельных корневых зон на этом скрытом сервере возможно частичное «выключение» Интернета в отдельных странах (на отдельных территориях). Частичное, потому что не все, например, русскоя-

зычные ресурсы находятся в доменах .ru или .su, их много и в общих корневых зонах .net, .com, .org и других. Особенно сейчас, когда корневых зон созданы сотни, на все случаи жизни.

Другими словами, корневые серверы являются распределенной и децентрализованной компьютерной системой, но это лишь транспорт, механизм доставки. Корневая зона DNS ими ретранслируется из единого центрального источника. Поэтому расхожее **мнение о децентрализованной и никем в одиночку не управляемой DNS,** которое часто можно встретить в интернетовских форумах, **в корне неверно**, на самом деле не всё так однозначно и всё под контролем.

Но есть и другой способ «выключения Интернет» — отключение протоколов маршрутизации: EGP, BGP, OSPF. Как раз отключением провайдерами поддержки протокола OSPF выключался Интернет в Египте в 2011 году во время «арабской весны» (см. пункт 4.1.4). Аналогичное проделывал Китай в некоторых своих провинциях (например, в июле 2009 года в Синьцзяне на целый год), вероятно, в качестве отладки «системы пожаротушения». В целом в 2018 году уже было 188 таких «выключений» Интернета в разных странах. Из свежих, нам достаточно «известных» — недавняя попытка в Белоруссии.

И кстати, президент США имеет право отключить Интернет по крайней мере в США на 120 дней в соответствии с «Protecting Cyberspace as a National Asset Act» (https://www.hsgac.senate.gov/imo/media/doc/CybersecurityOnePage[1]%20s.34801.pdf, https://www.hsgac.senate.gov/imo/media/doc/CybersecuritySection.pdf). По истечении 120 дней требуется продление через Сенат США.

И есть ещё третий способ отключения Интернета — самый-самый. Об этом способе автор вам обычно рассказывает на первой лекции по ОС, смотрите также начало Введения к данному учебному пособию. Этот способ заключается в том, чтобы некоторым образом добиться исчезновения unix/linux во всём мире. Специфика этого способа в том, что в отличие от предыдущих двух, в которых возможно восстановление Интернета «по звонку» от Папы в течении максимум нескольких часов (включив снова рубильник), в этом способе отключение будет тотальным и безвозвратным, точнее восстановит его можно будет только в лучшем случае лет через пять (а может и больше) при условии героического труда программеров и админов во всём мире. И вот в этом способе мы точно попадём вот сюда — см. рис. 34-5.

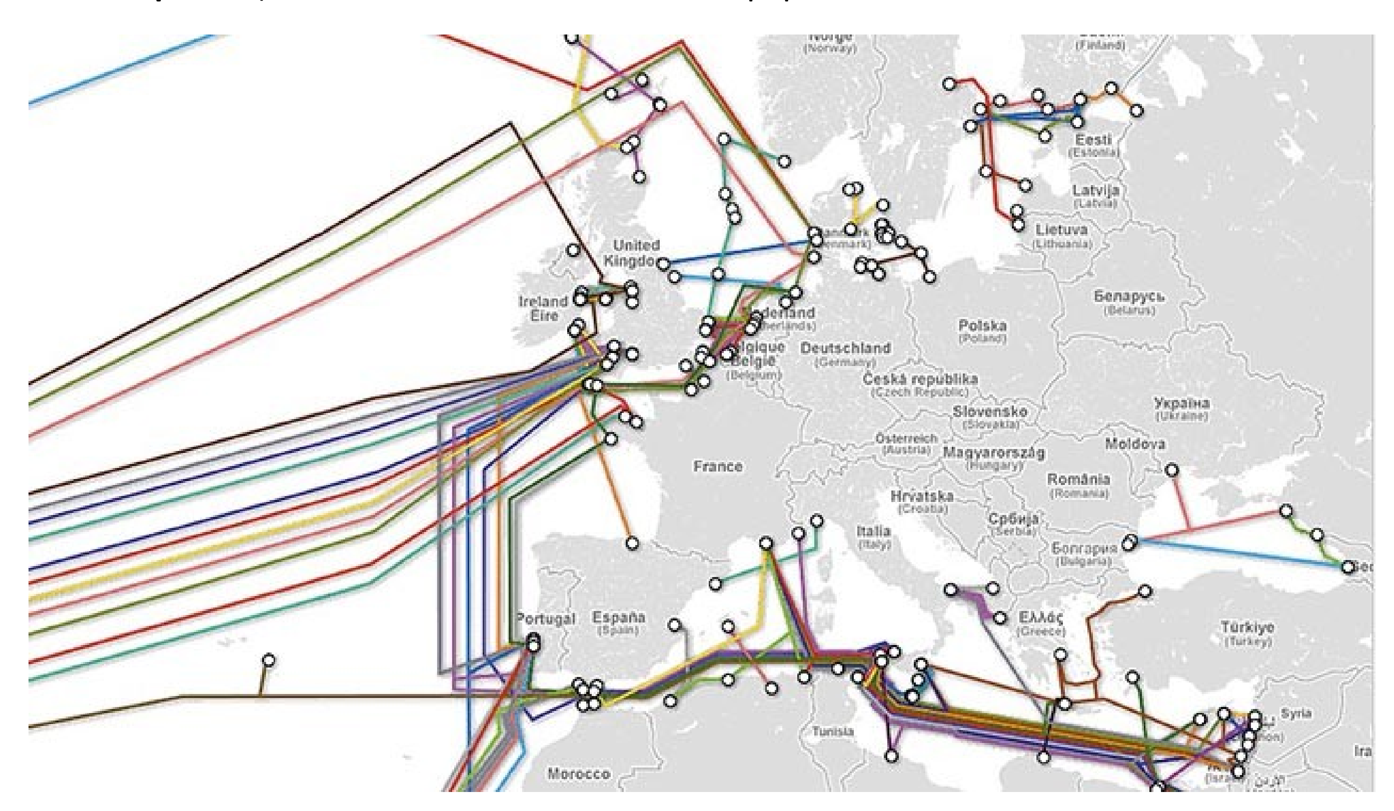



*Рис. 34-1.* Подводные кабели вокруг Европы по состоянию, примерно, на 2015 год



*Рис. 34-2.* Подготовка к укладке подводного кабеля



*Рис. 34-3.* Подводный кабель на дне. «Штучка» с болтами — грузило, чтоб не всплыл или не сдвинуло течением.

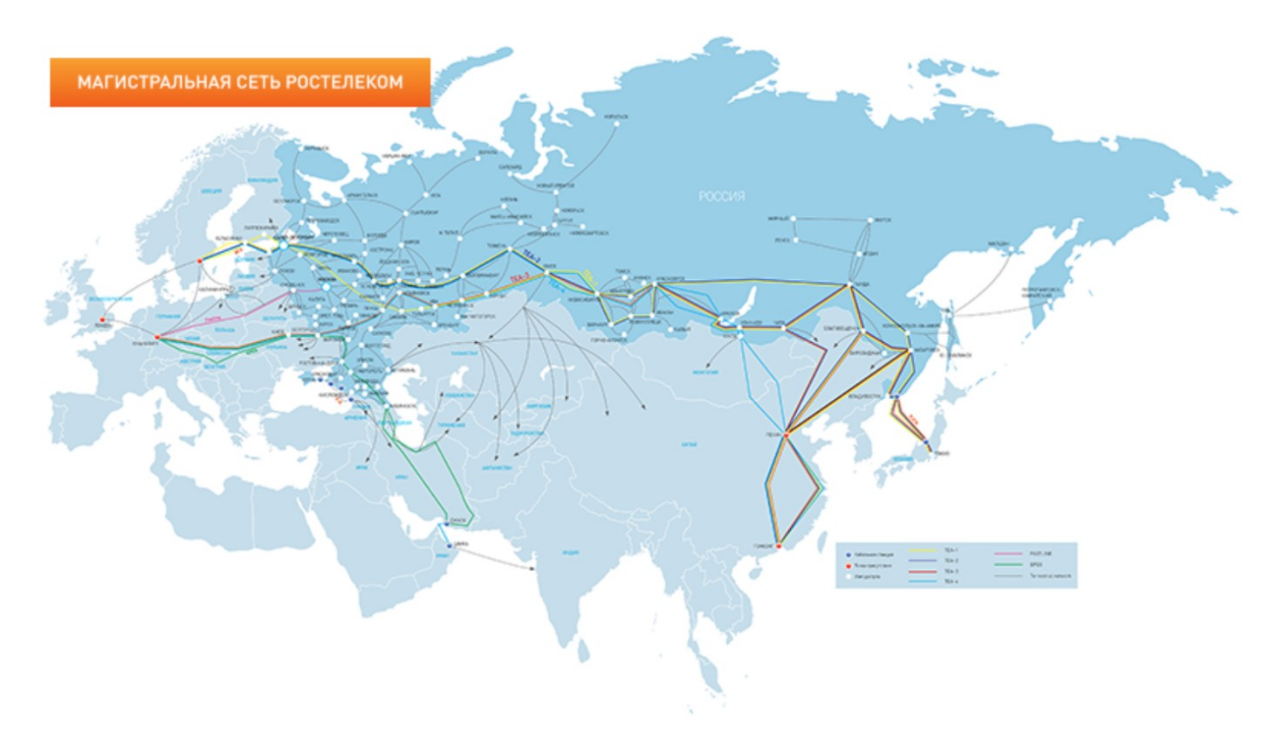



*Рис. 34-4.* Магистральная сеть Ростелеком



*Рис. 34-5.* Место, куда мы попадём, если во всём мире вдруг исчезнут unix/linux. Не шутка.

### 5.2.3. Первый уровень

Давным-давно (до 1985 года), когда Интернет был чисто «американской штучкой», на первом уровне были только домены org., mil., com., net., edu., gov., int., arpa. То есть, каждой из масштабных областей деятельности человечества достался свой домен. Коммерция, сети телекоммуникаций, некоммерческие объединения, власть, международная деятельность, образование и наука, ведение войн и подготовка к ним — все получили по домену.

Кроме того, был создан домен ARPA (Advanced Research Project Agency) — так раньше называлось Агентство перспективных разработок Министерства обороны США, в котором создали Интернет (в настоящее время оно называется DARPA — Defence Advanced Research Project Agency). То есть, создатели Интернета создали и для себя собственный домен, который служил чисто техническим целям. Служит он им и сейчас, играя важную роль в малоизвестных среди рядовых пользователей внутренних протоколах организации информационной структуры Интернета.

Затем, на рубеже 1980-90 годов, по мере приобретения Интернетом международной сущности, появились географические домены 1-го уровня:

us, ru, su, uk, ws и т. д. Их статус был закреплен международным стандартом ISO 3166-1. То есть, для каждого государства был создан свой домен 1-го уровня (см. рис. 36). Всего сейчас примерно 250 государств, соответственно и географических доменов примерно столько же. С одним уточнением: некоторые территории пока ещё не являются самостоятельными государствами, а имеют статус зависимых территорий (типа колоний), но и для них домены созданы.

**Пример:**

- .cc — Кокосовые острова — группа островов в Индийском океане, принадлежат Австралии пока;
- .fk — Фолклендские острова — британская заморская территория;
- .pf — Французская Полинезия — заморская территория Франции.

Или домены уже несуществующих государств: .dd — ГДР, .yu — Югославия.

Или альтернативный пример: .su — СССР, но домен по-прежнему активно используется.

Или даже так: .aq — Антарктика — никакого государства нет и даже не ожидается.

А потом, ещё через десять лет, политика была изменена (в связи с бурным ростом Интернета) и новые домены 1-го уровня посыпались, как из рога изобилия и сейчас их уже свыше полутора тысяч. В том числе и на национальных языках.

Замечание. Как обрабатываются имена на национальных языках, ведь пакет BIND обрабатывает только имена с весьма ограниченным алфавитом (см. пункт 2.4.3). Ответ: посредством приведения (перевода — алгоритм Punycode) имён из национальных алфавитов к разрешённому и передаче на обработку DNS уже имени в разрешённом алфавите. Выглядит такое «переведённое» имя уже не так красиво, как в национальном алфавите, но DNS его уже понимает.

Например, имя .рф преобразуется в имя .xn—p1ai.

Доменное имя первого уровня не может быть куплено частным лицом — оно является «общественным достоянием».

### 5.2.4. Второй уровень

Второй уровень — это уровень организаций и частных лиц. Все имена 2-го уровня (см. рис. 36) во всех доменах 1-го уровня кому-то принадлежат. Но временно. Типа, сдаются в аренду. Обычно на один год. Этот процесс передачи прав на имя называется регистрацией имени и является платной процедурой.

На верхнем уровне этого процесса работают так называемые «Региональные Регистраторы» — RIR, некоммерческие организации, подчиняющиеся ICANN и занимающаяся вопросами адресации и маршрутизации в сети Интернет (см. рис. 35). Они обеспечивают техническую сторону функционирования Интернета: выделяют IP-адреса, номера автономных систем, регистрируют обратные зоны DNS и решают другие технические вопросы. Также они имеют отношение к статистическому анализом сетей, мониторингу точек обмена трафиком и поддержке корневых зон DNS.

Статус RIR присваивается ICANN. Все RIR являются организациями, существующими на взносы своих членов. IANA (Root Zone Database — Internet Assigned Numbers Authority) делегирует RIR большие объёмы Интернет-ресурсов, которые RIR переделегируют своим членам в соответствии со своими правилами.

Все RIR коллективно образуют NRO (*Number Resource Organization*), созданную для представления интересов RIR и глобального взаимодействия.

На данный момент существуют пять RIR (см. рис. 35):

- American Registry for Internet Numbers (ARIN) — для Северной Америки;
- RIPE Network Coordination Centre (RIPE NCC) — для Европы, Ближнего Востока, Центральной Азии, постсоветского пространства;




*Рис. 35.* Региональные Интернет-регистраторы и зоны их ответственности

- Asia-Pacific Network Information Centre (APNIC) — для Азии и Тихоокеанского региона;
- Latin American and Caribbean Network Information Centre (LACNIC) — для Латинской Америки и Карибского региона;
- African Network Information Centre (AfriNIC) — для Африки и региона Индийского океана.

Ниже RIR находятся национальные регистраторы, сопровождающие прежде всего географические домены 1-го уровня. Но они же сопровождают в пределах национальных границ и все остальные домены 1-го уровня. То есть, для регистрации имени 2-го уровня, юридическому или физическому лицу необходимо обратиться к национальному регистратору. Но это правило нежёсткое и если очень хочется, то можно непосредственно выйти на уровень RIR для регистрации, только это не будет легче.

В России таким национальным регистратором является РосНИИРОС — Российский Научно-Исследовательский Институт Развития Общественных Сетей — www.nic.ru. Именно эта организация сопровождает зоны .ru, .su, .рф и др.

Ниже национальных регистраторов находятся прочие регистраторы — обычные коммерческие организации, выполняющие ту же работу, что и национальные регистраторы также на коммерческой основе с той лишь разницей, что в случае чего RIR (в данном случае RIPE) будет обращаться прежде всего к РосНИИРОС — о прочих регистраторах RIR может и не знать. Хотя с точки зрения клиента регистрация имени практически ничем не отличается.

Однако, есть географические домены 1-го уровня, созданные для стран, в которых в силу отсутствия возможностей (экономических, кадровых) нет национальных регистраторов. И тогда такие домены сопровождают кто-то иной, иногда некоммерческие организации, регистрируя имена 2-го уровня на очень свободных условиях.

Таким образом, имя 2-го уровня могут зарегистрировать различные организации и частные лица и в этом случае это имя будет представлять либо имя сетевого узла (так нередко делают частные лица), либо целую структуру нижестоящих имён — локальную или корпоративную сеть юридического или физического лица.

Как правило, организации-регистраторы, начиная с национальных регистраторов и ниже, помимо функции регистрации имён, также реализуют функцию сопровождения имён, то есть выступают в роли провайдеров Интернета — так это называется.

### 5.2.5. Третий уровень и последующие

Третий уровень именования и последующие (нижестоящие) (см. рис. 36) используется:

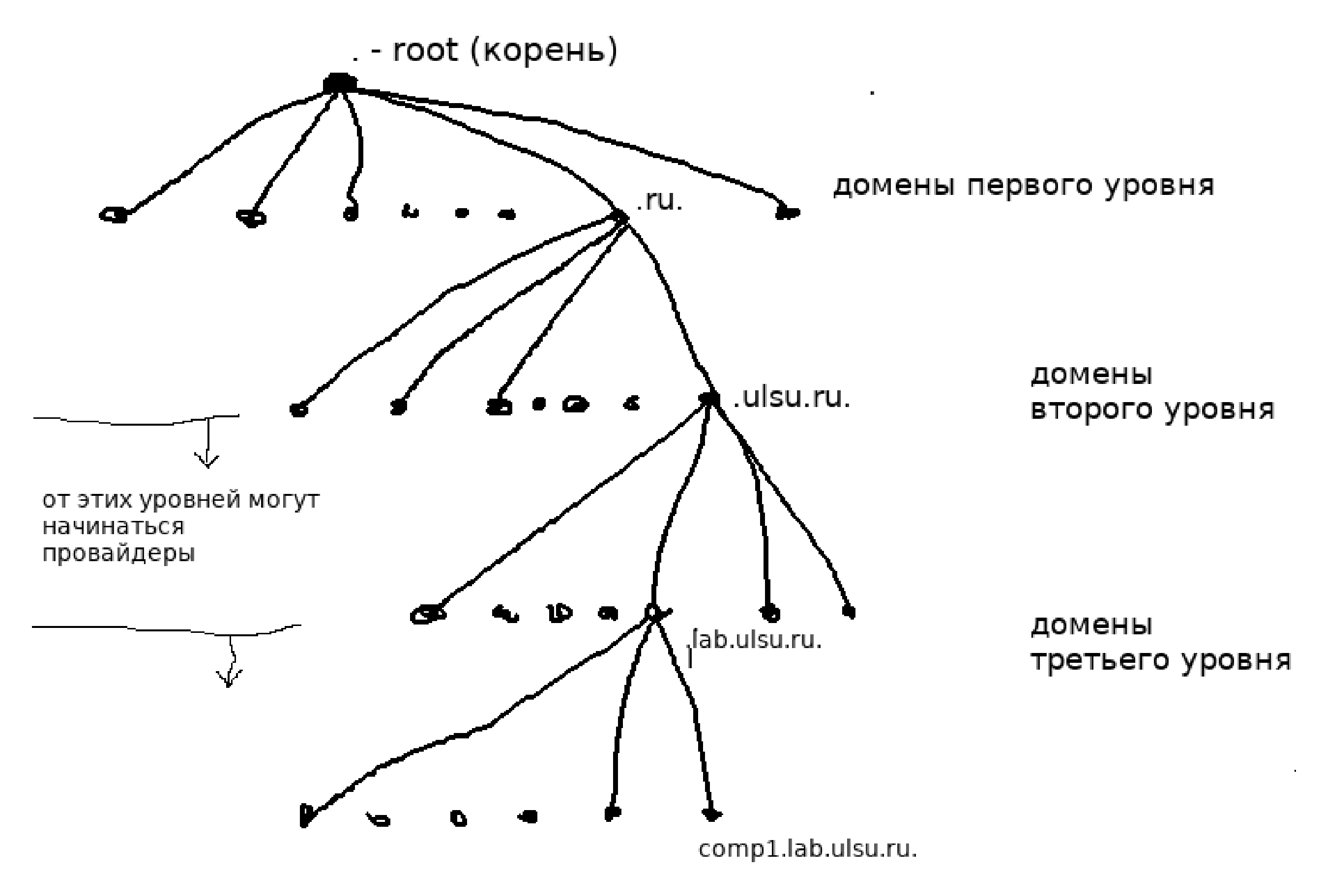



*Рис. 36.* Доменная структура интернет. Рисунок показывает, что пространство символьных имён имеет иерархическую структуру. Причём, (очень важно!): граф — не дерево, а сеть. На рисунке это не показано, но об этом будет сказано в конце лекции. Вопрос «на засыпку»: Чем отличается граф-дерево от графа-сети?

- для внутреннего именования сетевых объектов внутри сетей организаций (или вообще частных сетей), если организация владеет доменным именем 2-го уровня; в данном случае, «сеть организации» — это то, что мы обычно обзываем термином «корпоративная сеть» — см. примеры в пункте 2.5.1;

- для предоставления имён 3-го уровня другим организациям или частным лицам на коммерческой основе — это случай Интернет-провайдера.

Причём, имя третьего уровня и нижестоящих совершенно не обязательно где-либо регистрировать — это внутреннее дело юридического или физического лица, владеющего соответствующим именем 2-го уровня. Здесь взаимоотношения определяются соответствующим Гражданским Кодексом и иным законодательством, регламентирующим взаимоотношения «хозяйствующих объектов» в соответствующей стране.

## 5.3. Провай»деры Интернета

Таким образом, провайдеры Интернета — организации, основным бизнес-процессом которых является сопровождение символьного именования Интернета во всех его проявлениях, сопровождения полного жизненного цикла символьного именования — от создания имён (генерации, ввод в обращение) до ликвидации имён (вывода имён из обращения).

То есть, символьные имена и их сопровождение — предмет труда провайдера Интернета, на деятельности по сопровождению символьных имён провайдеры делают деньги. Как видно из рисунка 24, провайдеры начинаются там, где символьные имена становятся объектом купли-продажи, то есть, со 2-го уровня именования. Суть их деятельности заключается в том, что сопровождение символьных имён Интернета — это высокие технологии, требующие соответствующей квалификации, которой не только лишь все обладают.

Функции провайдеров с точки зрения символьного именования:

а) формирование для клиента (предоставление в аренду) доменного имени некоторого уровня (второго и ниже) «на период времени»; подразумевается, что клиент имеет локальную/корпоративную сеть, которую сформированное и полученное в аренду имя будет представлять; то есть, сеть клиента (сетевые узлы сети) может быть «видна» из Интернета, причём работу по обеспечению видимости доменного имени будет обеспечивать провайдер, а видимость сетевых узлов сети клиента будет обеспечивать клиент самостоятельно;

б) формирование для клиента (предоставление в аренду) имени хоста некоторого уровня «на период времени», если клиент имеет только лишь компьютер (сетевой узел);

в) предоставление возможности клиенту доступа к другим именам в Интернет — выход в пространство имён Интернет;

г) обеспечение возможности видимости предоставленного клиенту имени (возможно, провайдером же сформированного) в пространстве имён Интернет;

д) обеспечение возможности доступа из Интернет к хосту клиента (точнее, к запущенным на хосте сервисам);

е) обеспечение возможности доступа из Интернет к хостам локальной/корпоративной сети клиента (точнее, к запущенным в ней сервисам);

- и др.

## 5.4. Понятия URI, URN, URL

В Интернет для обращения к веб-документам изначально используется стандартизованная схема адресации и идентификации, учитывающую опыт адресации и идентификации таких сетевых сервисов, как e-mail, telnet, ftp и т.п. — URL, Uniform Resource Locator.

URL (RFC 1738) — унифицированный локатор (указатель) ресурсов, стандартизированный способ записи адреса ресурса в www и сети Интернет. Адрес URL имеет гибкую и расширяемую структуру для максимально естественного (ориентированного на человека) указания местонахождения ресурсов в сети. Для записи адреса используется ограниченный набор символов ASCII. Общий вид адреса можно представить так:

<схема>://<логин>:<пароль>@<хост>:<порт>/<полный-путь-к-ресурсу>

Где:

**схема** — схема обращения к ресурсу: http, ftp, gopher, mailto, news, telnet, file, man, info, whatis, ldap, wais и т.п.

**логин:пароль** — имя пользователя и его пароль, используемые для доступа к ресурсу

**хост** — доменное имя хоста или его IP-адрес.

**порт** — порт хоста для подключения

**полный-путь-к-ресурсу** — уточняющая информация о месте нахождения ресурса (зависит от протокола, например, для схемы file:// это будет путь к файлу в файловой системе).

**Примеры URL:**

http://example.com #запрос стартовой страницы по умолчанию
http://www.example.com/site/map.html #запрос страницы в указанном каталоге
http://example.com:81/script.php #подключение на нестандартный порт
http://example.org/script.php?key=value #передача параметров скрипту
ftp://user:pass@ftp.example.org #авторизация на ftp-сервере
http://192.168.0.1/example/www #подключение по ip-адресу
file:///srv/www/htdocs/index.html #открытие локального файла
gopher://example.com/1 #подключение к серверу gopher
mailto://user@example.org #ссылка на адрес эл.почты

В августе 2002 года **RFC 3305** анонсировал устаревание URL в пользу URI (Uniform Resource Identifier), еще более гибкого способа адресации,

вобравшего возможности как URL, так и URN (Uniform Resource Name, унифицированное имя ресурса). URI позволяет не только указавать местонахождение ресурса (как URL), но и идентифицировать его в заданном пространстве имен (как URN). Если в URI не указывать местонахождение, то с его помощью можно описывать ресурсы, которые не могут быть получены непосредственно из Интернета (автомобили, персоны и т.п.). Текущая структура и синтаксис URI регулируется стандартом **RFC 3986**, вышедшим в январе 2005 года.

Однако, в данном пособии этот вопрос — именование специализированных ресурсов — не рассматривается.

**Вопросы «на засыпку»**

1. Если клиент подключен «по выделенке», то как будут выглядеть функции провайдера по отношению к этому клиенту?

2. Разбор доменного имени производится: слева направо, справа налево или по желанию программера-разработчика приложения?

3. Как называется совокупность подключенных к сети компьютеров, принадлежащих какому-либо лицу или организации и имеющая уникальное зарегистрированное имя?

4. Поскольку все адреса в интернете объединены в глобальную сеть, то доменная система имен должна обладать свойством . . . (каким?).

5. Является ли имя www.ulsu.ru доменjv второго уровня?

6. Как называется система доменных имён, обеспечивающая возможность использования для адресации узлов сети символических имён вместо числовых адресов?

7*. Для реализации каких функций используется домен ARPA?

8*. В чём разница между граф-дерево и граф-сеть? В вопросе имеются в виду виды графов: чем отличается граф типа «граф-дерево» от графа типа «граф-сеть»?

# Лекция 6. Стек TCP/IP. Сервисы

## 6.1. Определения и содержания понятий

### 6.1.1. Определение и смысл сервиса

*Определение 1.* **Сервис** — это аппаратно-программный комплекс, предназначенный для организации публичного доступа к некоторому ресурсу (контенту) — см. рис. 37. В обиходе, термин «организация публичного доступа» означает «расшаривание» этого ресурса.

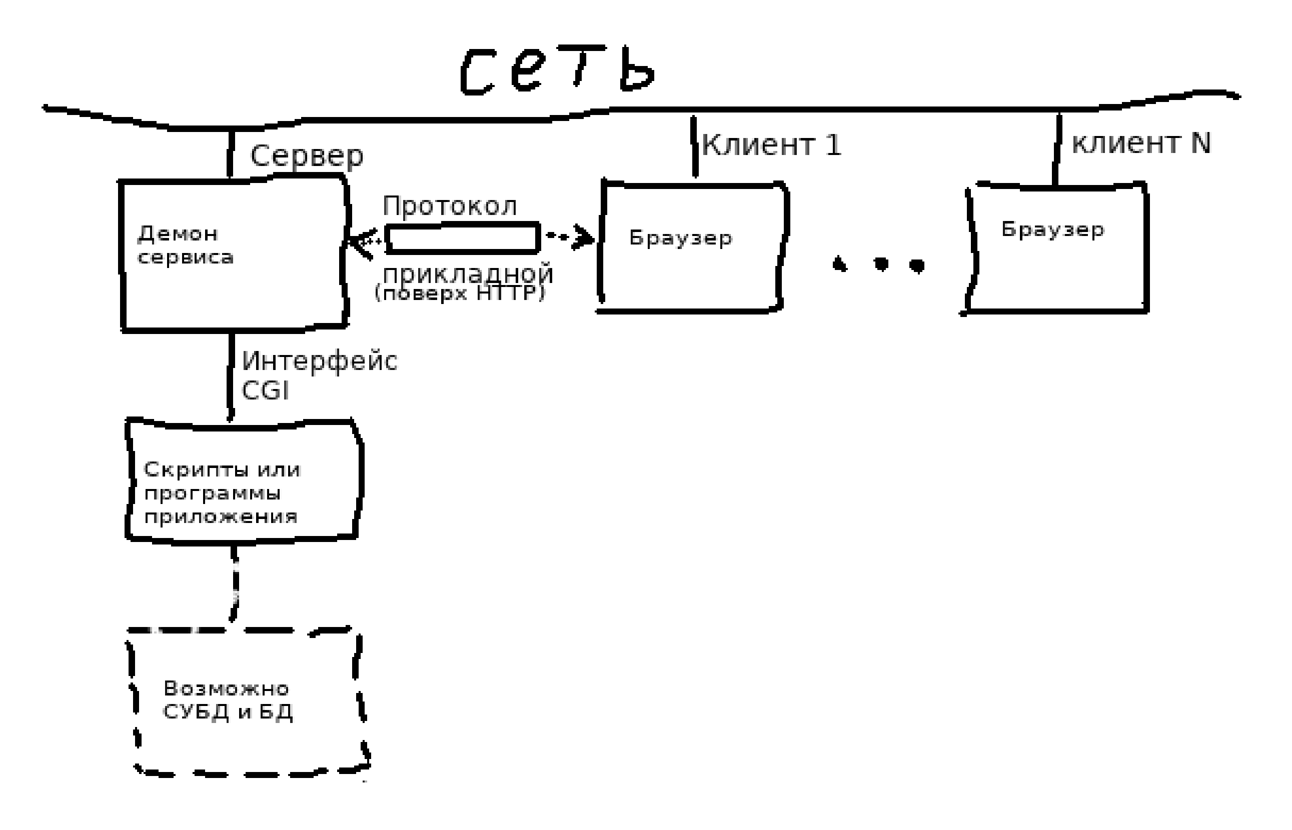



*Рис. 37.* Сервис — это аппаратно-программный комплекс, предназначенный для организации публичного доступа к некоторому ресурсу (контенту)

На рисунке 37 показаны составляющие сервиса:

- *клиенты сервиса* — аппаратно-программный комплексы, генерирующие запросы на получение доступа к контенту сервиса;

- *субъект сервиса* (он же «демон» сервиса) — аппаратно-программный комплекс, выполняющий функцию посредника между клиентами сервиса и контентом сервиса; его назначение: принять запрос от клиента, сформировать ответ, обратившись к контенту, отправить ответ клиенту;

- *протокол сервиса* — правила, определяющие порядок взаимодействия клиентов сервиса субъекта сервиса;

- *объект сервиса* — контент сервиса — та информация, что хранится в БД сервиса либо те сущности, к которым предоставляется доступ (например, вычислительная мощность или место для хранения); может иметь весьма разнообразное содержание, прежде всего и чаще всего это информация различного вида и различным образом организованная, в старину именно она и была контентом; однако в настоящее время контент стал весьма разнообразным: услуги различного вида, вычислительная мощность, память для хранения информации, доступ к программам и т. д.;

- *предмет сервиса* — то специфическое представление контента, в которое субъект сервиса преобразует информацию, или тот специфический способ доступа к контенту, который предоставляет субъект сервиса;

- *метод сервиса* — алгоритмы преобразования контента сервиса в предмет сервиса, метод сервиса реализуется скриптами, подключаемыми к демону по интерфейсу CGI.

Для примера «из жизни» смотрите понятие «лоббизм» (https://ru.wikipedia.org/wiki/Лоббизм — «Структура лоббирования: объект, субъект и предмет»). Лоббирование предполагает наличие следующих составляющих:

- *регламентирующее законодательство*, определяющее правила взаимоотношений объектов лоббистской деятельности;

- *клиент лоббирования* — физическое или юридическое лицо, желающее повлиять на разработку, принятие и осуществление должностными лицами и депутатами органов законодательной власти законодательных актов, политических решений, подзаконных нормативных актов, административных решений и т. д.;

- *субъект лоббирования* — лоббист (агент «группы давления») то есть профессиональный **посредник** между клиентом и объектом лоббирования. Лоббистом может быть физическое или юридическое лицо. При этом лоббист может работать либо за гонорар, либо быть получающим зарплату сотрудником компании-клиента. Например, в США, где лоббистское законодательство очень детальное, лоббистом может быть отдел крупной корпорации, специализирующийся на лоббизме;

- *объект лоббирования* — государственные органы и органы местного самоуправления, а также должностные лица этих органов. Кроме того, объектом лоббирования могут быть орган или должностное лицо международной организации (например, Евросоюза);

- *предмет лоббирования* — цели, которые ставят перед собой лоббисты при оказании давления на орган государственной власти;
- *методы лоббирования* — технологии, используемые при продвижении интересов в органах государственной власти.

Таким образом, смысл сервиса в том, что «пряников всегда не хватает на всех» (С), поэтому нужно предпринимать некоторые правильные организационные меры по наведению порядка среди жаждущих (организация толпы в очередь) и формирования правил (ограничений) на получение «пряников». В противном случае, система может просто пойти в отказ из-за перегрузки — пример: атака dDOS. Или в примере с лоббизмом — зашкаливающая коррупция.

*Определение 2.* **Сервис** — это некоторая функциональность системы, доступная нескольким (всем) процессам/пользователям в многозадачной многопользовательской системе.

*Определение 3.* **Сервис локальный,** если он доступен только процессам в локальной системе, то есть, в пределах одной ОС. Обратите внимание: не в пределах одного компьютера, а именно в пределах одной ОС, то есть, виртуальная система — это уже другая ОС.

*Определение 4.* **Сервис сетевой,** если он доступен в сети, то есть, с других ЭВМ сети, и неважно, какая это сеть — локальная, или корпоративная, или вообще Интернет.

**Следствие**. Сервис определяется только для многозадачных систем. В однозадачных системах это понятие существовать не может (причина: последовательное выполнение процессов), поскольку понятие сервиса требует как минимум псевдопараллельности. Если более точно, то на однозадачной системе может быть создан только один сетевой сервис и только он.

**Замечание.** Здесь определяются и ниже описываются сервисы, а не какие не службы! Служба — это совсем другое. На примерах: армейская служба — работа по защите отчества; «ходить на службу» (чиновники ходят на службу, на работу); служба в церкви/храме/мечети — бывали в церкви? (сейчас это модно), вот, вы ходили на церковную службу и батюшка эту службу вёл, работал; батюшка в церковь ходит на службу (на работу), а вы ходите в церковь совсем за другим (во-первых — отъюстировать свою психику, во-вторых — напосмотреть). И т. д. И вообще: «служить бы рад, прислуживаться тошно» (С) — здесь как раз отделяется понятие «служить» (службу, выполнять работу) от понятия «прислуживаться» — предоставлять услугу-сервис, то есть, прислуживающийся предоставляет ин-

терфейс хозяину, по которому хозяин слугу имеет. То есть, сервис — это процесс предоставления услуги, а услуга — это, что потом выполняется демоном и тем, что за ним стоит, то есть, работа по подготовке ответа.

А то, что MS везде использует термин «служба» — это от незнания русского языка.

### 6.1.2. Назначение сервиса

Назначение сервиса — предоставить клиентам простой, унифицированный (всегда одинаковый) и надёжный доступ к контенту. Для обеспечения этого в сервисе выделяются функциональные составляющие, которые качественно выполняют свои узкие функции: демон — быстрая обработка запросов, СУБД — надёжное хранение информации, бизнес-логика — преобразование информации.

Для разъяснения ситуации ещё один пример.

Предположим, есть склад с тостами (которые хлебобулочные изделия). Предположим, есть ещё один склад с сырной нарезкой. И есть много жаждущих/голодных, которые желают себе сделать пару бутербродов с сыром. Первый из них подходит к складу с тостами и начинает выбирать себе тосты, блокируя при этом доступ на склад для других (монопольный захват ресурса — склад большой, пока выберешь . . .). Другой жаждущий, обнаружив, что склад с тостами не доступен, решает пока выбрать себе сырную нарезку для пары бутербродов, также блокируя при этом склад сырных нарезок (опять монопольный захват ресурса — склад большой, пока выберешь . . .). Первый жаждущий, выбрав себе первый тост, пытается заполучить сырную нарезку со второго склада, чтобы проверить на практике, насколько это вкусно будет выглядеть, но обнаружив, что склад с сырной нарезкой недоступен, переходит в состояние ожидания, не разблокируя при этом склад тостов. Второй жаждущий, выбрав себе первую сырную нарезку, ждёт доступа в склад тостов, чтобы оценить, будут ли соответствовать его ожидания вкусовых качеств запланированного бутерброда с его внешним видом и, поскольку склад тостов заблокирован, также переходит в состояние ожидания, не разблокируя при этом склад сырных нарезок. Остальные жаждущие/голодные вообще остаются при своих интересах. Тупик. Система распределения «пряников» практически «повисла».

Решение. Перед складами ставим столы с пультом управления автоматизированной складской системой и садим за них столоначальников-менеджеров — исключительно для контроля (параноики мы — не доверяем автоматике), ну и для юсеробильности (юзаби́лити — *usability,* дружественный интерфейс — резиновая накладка на ручку граблей; вдруг тётя

придёт и заявит: «Да не умею я ! . .» и тогда столоначальник подбежит и популярно объяснит). Для клиентов ставим терминалы заказов (см. рис. 38). Результат: клиент подходит, выбирает на терминале нужные ему тосты, складская система (она же автомат — всё помнит, где что лежит и работает быстро) тут же выдаёт заказ, клиент идёт на второй склад за сырной нарезкой, где также автомат быстро выдаёт заказ — самое то для голодующих, фастфуд, называется.

Терминал заказов — демон сервиса. Склад — база данных с контентом. Складская автоматизированная система — скрипты бизнес-логики. Теперь клиент тратит минимум времени и практически никаких очередей. Естественно, в пределах пропускной способности. И всё потому, что клиента лишили возможности монопольного захвата ресурса — создали систему «расшаривания» ресурса, то есть, сервис.

Ну, уж теперь-то после столь подробного объяснения с примерами совсем понятны определение, смысл и назначение сервиса — авторы надеются.

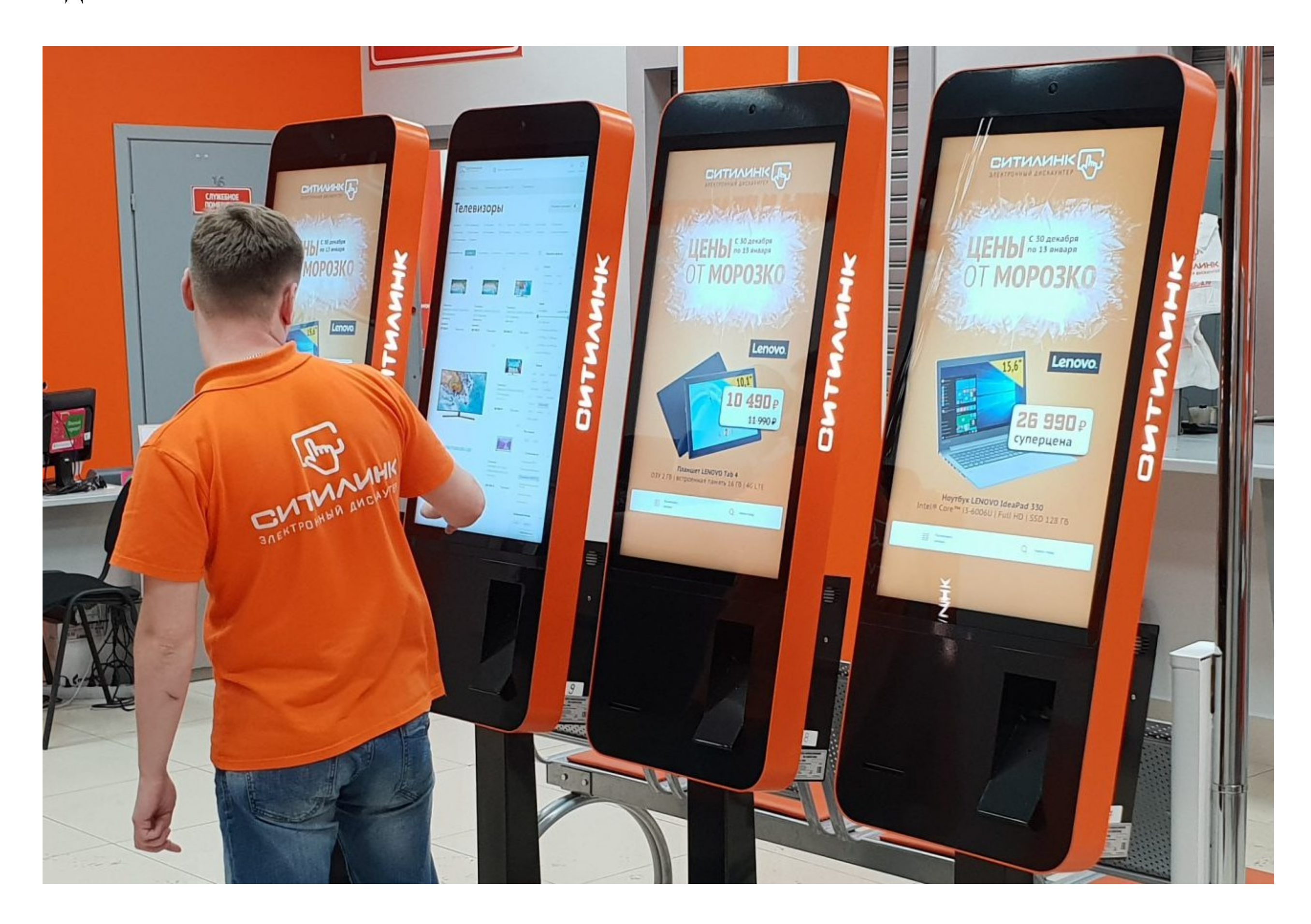



*Рис. 38.* Терминалы заказов в интернет-магазине «Ситилинк» — не реклама, а пояснение назначения сервисов

### 6.1.3. Виды сервисов

Сервисы бывают локальные и сетевые.

**Локальные сервисы (не сетевые)** — очень распространены, но малоизвестны. Настолько, что «просто пользователь» даже затруднится пример привести такого сервиса. Хотя примеры очевидны и это прежде всего операционная система в целом — локальный сервис, обеспечивающий доступ системному и прикладному программному обеспечению к ресурсам компьютера, причём, этот сервис очень многогранный, типа мультисервисный, включает в себя очень много возможностей. Кроме того, в компьютере работают ещё десятки различных сервисов, как то:

- сервис времени, расшаривающий микросхему таймера,
- сервис syslog, расшаривающий функцию ведения логов (протоколов работы),
- сервис systemd-journald, ведение журнала,
- сервис systemd-udevd, отслеживание устройств и их именования,
- сервис acpid, доступ к питанию компьютера,
- и т. д.

**Сервисы сетевые** — доступные из сети. Когда говорят про сервисы, практически всегда имеют в виду именно такие сервисы. Это сервисы www, e-mail, ftp, torrent, ICQ, telnet и ssh, видеоконференции/вебинары — дальнейшее расширение сервиса www в область интерактивного взаимодействия, SaaS — предоставление доступа к software как к сервису (сдача в аренду ПО), и даже PaaS и IaaS — платформа и инфраструктура, как сервис. Социальные сервисы: vk.com («в контакте»), ok.ru («одноклассники»), fb.com (facebook.com — очень гадкая вещь: банят, козлы, не глядя), t.me (telegramm.org) и др. Социальные видеосервисы: youtube.com (видеоразвлекалочка, но, аналогично, очень гадкая вещь: банят не глядя), rutube.ru (тоже развлекалочка, но ограниченная), instagram.com и т. д.

И то, что мы имеем сейчас в области сетевых сервисов — далеко не последнее слово в сервисинге. То ли ещё будет. Примерное будущее: Apple, Google и Microsoft, которые пользуясь своим монополизмом в определённых областях, уже организовали тотальную слежку за нами. На эту же цель «намылились» Samsung и Huawey. Авторы вам не завидуют, авторы старенькие и есть надежда — не доживут до реализации высокотехнологичных задумок.

## 6.2. Состав и структура сервиса

### 6.2.1. Структура сервиса

Сервис реализуется некоторой программой или комплексом программ, работающим в «автоматическом режиме» («режиме демона») — без диалога с оператором/пользователем (почти всегда). То есть, сервис — это некоторая программа (процесс), запущенная на ЭВМ, и которая по запросу от других программ (процессов), действующих, возможно, по командам пользователей, выполняет некоторые действия, некоторую работу, которая либо недоступна процессам-заказчикам (если сетевой сервис), либо является общей и потому вынесена в отдельный общедоступный процесс (если локальный сервис).

Иначе говоря, **сервис всегда работает в архитектуре «клиент-сервер»**.

На рисунке 39 показана структура сетевого сервиса.




*Рис. 39.* Структура сетевого сервиса

Структура локального сервиса отличается только тем, что протокол сервиса реализуется не через сеть, а через локальные средства межпроцессного взаимодействия: локальный сокет (очень часто), именованный (тоже часто) или неименованный (редко) каналы. То есть, на рисунке вместо слов «Сеть локальная или глобальная» будет написано, например, «Механизм локальных сокетов». На рисунке 39 обведены штриховой линией те элементы, которые составляют непосредственно сервис. Клиенты в состав сервиса не входят, это отдельные процессы, возможно расположенные у чёрта на куличках.

Что мы видим на рисунке:

а) сервис имеет модульную структуру; сложные сервисы — это комплексы программ, как правило, это сетевые сервисы; в простейшем виде сервис состоит (если сервис локальный) только из расшариваемого ресурса, демона и протокола; пример: локальный сервис времени — очень простой демон + контент — микросхема таймера + протокол доступа к демону; сетевые сервисы сложнее даже в простейших случаях: пример, web-сервис со статическим сайтом — apache (весьма сложная программа) + каталог с файлами готовых html-страничек + протокол http;

б) элементы сервиса достаточно независимы друг от друга, то есть связаны только протоколами и интерфейсами;

в) реализация каждого сервиса — специфическая и определяется тем, что мы расшариваем и как мы это расшариваем;

г) специфика интерфейсов:

- интерфейс аппаратуры определяется разработчиком аппаратуры,

- интерфейс драйвера — типовой (обычно) и определяется правилами программирования модулей операционной системы,

- интерфейс файловой системы — определяется разработчиками файловой системы, как правило, это те же люди, что разрабатывали ОС,

- интерфейс подключения внешних программ определяется разработчиками демона,

- протокол сервиса специфичен для каждого сервиса и определяется разработчиками демона;

д) и как уже сказано выше, **сервис реализуется всегда в архитектуре «клиент-сервер»**,

е) из специфичности протокола сервиса следует, что клиенты сервиса также оригинальны для каждого сервиса; однако, бывают исключения: примеры — браузер, putty, то есть, существуют клиенты, которые поддерживают сразу несколько протоколов (разных сервисов);

ж) человек (пользователь) — только за клиентом сервиса, в самом сервисе его нет.

Перечисленные пункты а) — ж) точно и конструктивно определяют сущность сервиса.

### 6.2.2 Сервисы и стек TCP/IP

Использование сервисов практически всегда подразумевает использование unix`ового стека сетевых протоколов TCP/IP. Существуют реализации сервисов на базе некоторых других стеков сетевых протоколов. Однако авторы сомневаются, что вы с ними встретитесь хоть раз за свою долгую жизнь.

Это обусловлено тем, что

- в настоящее время глобальные сети реализованы на основе стека TCP/IP и никаких изменений этого положения не просматривается,

- сервисы в реализациях существенно задействуют технологии стека TCP/IP.

### 6.2.3. Сервисы и архитектура SOA

Невооружённым глазом видно, что рисунок 39 показывает типичную архитектуру SOA.

Так оно и есть. Термин «SOA-архитектура Информационных Систем» отсюда и произошёл.

Также обратите внимание, что в основе SOA-архитектуры ИС могут лежать разные сервисы, а совсем не обязательно веб.

Пример: в 90-ые годы были распространены в фирмах/организациях (и сейчас ещё встречаются) информационно-управляющие системы на основе почтового сервиса e-mail. Причина создания таких ИУС заключается в том, что почтовый сервис детально документирует процесс прохождения писем: создание, пересылку, получение, прочтение — всё фиксируется в протоколе работы почтовой системы с точностью до секунды, не отвертишься, что не знал: открыл письмо, значит прочёл, ознакомлен, ценное указание получил. То есть, то, что почтовая система ведёт очень подробный протокол своей работы — существенно облегчает создание на основе этого сервиса информационной системы с архитектурой SOA. К тому же сервис mail — лёгкий, слабонагружающий вычислительную систему, что в 90-е годы было существенным фактором.

## 6.3. Проектирование и программирование сервисов

### 6.3.1. Порядок разработки сервиса

Разработка сервиса делится на
- разработку демона,
- разработку протокола,
- разработку всего остального.

### 6.3.2. Прежде всего обо всём остальном:

- реализация клиента — он создаётся по правилам разработки сетевых программ, особой специфики нет, но есть два важных момента: а) программа, всё-таки, сетевая, б) интерфейс пользователя, который может состоять из двух составляющих — интерфейс самой клиентской программы + интерфейс, получаемый по сети (frontend);

- реализация «преобразования данных в распознаваемый клиентом вид» — скрипты (как в web-сервисе) или обычные программы командной строки («фильтры») — backend, особой специфики нет;

- разработка драйвера осуществляется по правилам программирования драйверов в данной ОС, особой специфики нет, но очень сложно, не только лишь каждый это может делать, в лучшем случае один из тысячи программеров может это делать.

### 6.3.3. Разработка протокола сервиса

В протоколе сервиса (протоколе информационного взаимодействия клиент-демон) должно быть определено:

- формат пакетов, которыми будут обмениваться клиент и сервер, с описанием полей пакета;

- алгоритм обмена: как правило, обмен инициирует клиент и простейший алгоритм — «старт-стопный» (запрос — ответ; см. новую технологию создания сервисов REST); если же запросы предполагаются сложными (с уточняющими запросами) и многопакетными отвименование ветами, то есть, предполагается «диалог» между клиентом и сервером, то необходимо построить автомат алгоритма, чтобы показать, что отсутствуют возможности для перехода в тупиковые состояния (зависания);

- кодирование информации — это важно для сетевых сервисов, ведь клиенты могут работать в разных операционных системах со своими понятиями о кодировках информации;

- именование взаимодействующих объектов — каким образом клиент найдёт сервис и какая адресная информация должна присутствовать в запросах/ответах и в пакетах данных.

Протокол сервиса — это протокол прикладного уровня и работает, как правило, поверх протоколов сетевого или транспортного уровней стека.

Примеры:

- протокол ftp — для сервиса ftp (демон ftpd, wsftpd и др. — клиенты lftp, mc, браузер и др.); протокол работает поверх протоколов TCP и UDP и включает в себя протокол telnet; — протокол telnet — для сервиса telnet (демон telnetd — клиент telnet); протокол работает поверх протокола TCP;

- http — для сервиса web (демон apache и др. — клиент браузер); протокол работает поверх протокола TCP;

- smtp и pop — для сервиса почты (демоны sendmail, popa3d, imapd и др. — клиенты mail, Thunderbird, Geary, Evolution и др.); протоколы работают поверх протоколов TCP и UDP;

- и т. д.

Но если разрабатывается Информационная Система в архитектуре SOA, то для взаимодействия клиента и демона должен быть определён (не разработан, а определён!) протокол прикладного уровня. Пример: интернет-магазин, форум и другие ИС на основе веб-сервиса подразумевают использование протокола http в качестве протокола ИС (протокола интернет-магазина или протокола форума). Однако при разработке подобной Информационной Системы оговариваются детали взаимодействия по этому протоколу, обусловленные тем, что в качестве данных в пакетах протокола http может пересылаться специфическая для данной ИС информация и, соответственно, должно быть оговорено, как её отправитель будет помещать в пакеты протокола http и как получатель будет распознавать и обрабатывать.

### 6.3.4. Разработка демона

Прежде всего, определение демона.

**Определение**. **Демон** — это существо, выполняющим задачи, за которые не хотят браться «боги», то есть, ещё не «бог», но около того, типа «ангела-хранителя».

**Определение**. **Демон** (daemon, dæmon, др.-греч. δαίμων божество) — компьютерная программа в системах класса UNIX, обычно запускаемая самой системой и работающая в фоновом режиме без прямого взаимодействия с пользователем. Обычно демоны большую часть времени проводят в ожидании некоторого события. Когда это событие происходит, демон активизируется, выполняет свою работу и снова засыпает в ожидании события: как «старик Хоттабыч» — сидит в кувшине и ждёт, пока его потрут. Вывод: найденный кувшин обязательно надо потереть, добрый демон — это голубая мечта любого разгильдяя.

Здесь очень важно уточнение о том, что программа демон не имеет взаимодействия с пользователем (диалога с пользователем), говорят: «не имеет управляющего терминала».

Настройка демона осуществляется двумя способами:

- через ключи (опции), указываемые в строке запуска;

- с помощью конфигурационных файлов, имя которых может быть указано с помощью ключа в строке запуска, либо задаётся по умолчанию — в большинстве случаев.

Очень редко возможности по управлению сервисом (точнее, демоном сервиса) включаются в протокол сервиса и тогда интерфейс клиента включает административные возможности. Конфигурационные файлы демонов обычно помещаются в каталог /etc или в его подкаталоги.

Отсюда следуют первые два требования к этой программе:

**а)** демон — это программа командной строки и, следовательно, она должна уметь обрабатывать ключи запуска; то есть, функция main должна определяться так:

int main (int argc, char *argv[])

то есть, она должна возвращать «код возврата» от операторов return N; находящихся где-то в теле функции, что бы скрипт запуска мог проверить результат запуска;

**б)** эта программа в начале своей работы должна уметь прочитать свой конфигурационный файл и настроиться на указанный в нём режим работы.

Кстати, ключи запуска, указываемые в командной строке, более приоритетны и должны переопределять установки из конфигурационного файла.

Следующее требование к разработке демона следуют из самой сущности этой программы:

**в)** она обрабатывает запросы и должна это делать как можно быстрее; иначе говоря, в ней должно быть выделено то ядро, что будет обрабатывать запросы, и эта часть программы должна быть написана как можно более эффективно с оптимизацией по скорости.

Парадигма демона представлена на рис. 40.

Как видно из рисунка, демон имеет в своём составе ряд действий, которых нет в обычных программах и которые, что бы их правильно сделать, требуют от программера существенного понимания как операционной системы, так и технологии программирования. Кроме того, центральной частью демона является «вечный цикл» do { . . . } while (1); , выход из которого осуществляется только по получении соответствующих особо ценных указаний.

```
Начало
Определения типов, переменных, функций;
 . . .
 int main(int argc,char *argv[])
{
 . . .
Обработка конфигурационного файла и ключей;
 . . .
Проверка отсутствия себя;
Определение рабочего каталога и маски;
Переключение в режим демона;
 — Создание родственного процесса;
 — Запись pid-файла;
 — Переопределение стандартных файлов;
 — Если необходимо — переход в системного пользователя;
 . . .
Определение параметров протоколирования;
Определение порядка слежения за процессом демона;
Определение и настройка интерфейсов и протоколов;
. . .
do {

 Приём запроса;
 Протоколирование запроса;
 Выполнение действий по запросу;
 Формирование и отправка ответа;
 Выполнение административных действий, если сигнал;

} while (1);

Организация выхода, «приборка мусора» за собой;
return 0;
}
Конец
```

*Рис. 40.* Парадигма демона

Так же следует отметить, что программирование демонов несколько различается для разных ядер linux.

**Вопросы «на засыпку»**

1. Возможно ли создание информационно-управляющей системы на основе сервиса почты?

2. Является ли сервисом «расшаривание» каталога в локальную сеть?

3. Форум — это не ограниченная временем система обмена информацией на определенную тему между пользователями сети. Является ли форум сервисом?

4. Является ли сервисом доступ к видеокарте для отрисовки изображений, предоставляемый драйвером видеокарты?

5. Является ли сервером комп с установленной на нём операционной системой linux?

6. Является ли сервисом имя:
http://www.vasya.com/dokument/spok.doc?

7. Является ли обращение к сервису следующее имя:
file://mmm.vasya.com/dokument/text.doc?

# Лекция 7. Стек TCP/IP. Управление сервисами

## 7.1. Методы запуска сервисов

### 7.1.1. Разовый запуск — «вручную»

Обычно запуск демона «вручную» осуществляется в отладочных или учебных целях. И здесь нужно выделить два случая:

- демон написан правильно (см. п. 6.2.);

- демон ещё не демон или это вообще не демон, но нужно запустить как демон.

Разница этих случаев в том, что в первом случае программа написана правильно и сама всё, что нужно сделает (см. парадигму демона), а во втором случае мы ещё недописали демона или вообще просто хотим запустить некоторую программу как фоновый процесс.

Запуск в первом случае:

# progad <ключи> <Enter>

то есть, никаких особенностей, запускаем как обычную программу, но от root'a.

Запуск во втором случае:

$ proga <ключи> & <Enter>

то есть, символом «&» специально указываем системе, что мы хотим получить «отсоединённый от терминала» фоновый процесс, потому как наша программа ещё не умеет, или вообще не умеет самостоятельно переходить в режим фонового процесса (демона). И в этом случае, как правило, запуск осуществляется от обычного пользователя. Однако если мы запускаем в отладочных целях ещё недоделанный демон, то нужно запускать от root'a.

### 7.1.2. Схема BSD

В этой схеме для запуска демона сервиса и в целом всего сервиса должен существовать «скрипт запуска» сервиса. В этом скрипте выполняется некоторые предварительные действия:

- назначаются нужные переменные среды и проверяются их значения;

- проверяется наличие нужных программ из состава сервиса и пути к ним;

- если необходимо, создаётся структура рабочего каталога сервиса и кладутся туда нужные файлы;

- определяются названия некоторых служебных файлов сервиса и пути к ним;
- определяется название сервиса (как он будет обзываться в системе), оно часто не совпадает с именем демона;
- определяются «точки входа» в скрипт: start/restart/stop/status/reload и др.;
- формируются ключи командной строки для запуска демона;
- если сервис — комплекс программ, то запускаются нужные в определённом порядке;
- проверяются, запущены ли сервисы, от которых зависит данный;
- и т. д., то есть, выполняются те работы, которые нежелательно в целях обеспечения нужной масштабируемости, переносимости и гибкости настройки выполнять непосредственно в демоне.

Этот скрипт помещается в каталог /etc/rc.d/, ему назначается хозяин root и назначается режим доступа 755, то есть, ставятся биты исполнения. Далее, или непосредственно в основном конфигурационном файле rc, или в rc.local ставится вызов этого скрипта. Если система настроена так, что при запуске в конце конфигурирования она просматривает каталог rc.d на предмет поиска исполняемых скриптов и запуска их, то тогда вставлять вызов скрипта в файлы rc нет необходимости, достаточно просто сделать скрипт исполняемым.

Сложность в этой схеме в том, что администратор должен самостоятельно отслеживать правильный порядок запуска сервисов в системе: либо давая соответствующие имена скриптам, чтобы выстроить их в нужном порядке в каталоге /etc/rc.d/ (отсортировать), либо вставить вызов скрипта в файлы rc в нужное место. То есть, к администратору предъявляются повышенные требования — он должен понимать, что делает и уметь программировать на shell.

### 7.1.3. Схема SystemV

В этой схеме также для запуска сервисов используется аналогичные стартовые скрипты, но они помещаются в каталог /etc/rc.d/init.d/ — «свалка» скриптов. Дополнительно в системах, настраиваемых по схеме systemV, создаются «уровневые» каталоги, которые располагаются также в /etc/rc.d/ и называются rc0.d, rc1.d, rc2.d. rc3.d, rc4.d, rc5.d, rc6.d и которые определяют то, как должна быть настроена система при выходе на нужный уровень работы. Сам уровень работы определяется в файле /etc/inittab — называется «The default runlevel»:

0 — выключение системы,

1 — однозадачный однопользовательский режим, обычно для исправления ошибок,

2 — многозадачный многопользовательский, но без поддержки сети (не всегда),

3 — нормальный полнофункциональный режим без графики — серверный,

4 — то же, что третий, считается «промежуточным», обычно не используется,

5 — нормальный полнофункциональный режим с графикой, «десктопный»,

6 — перезагрузка,

В «уровневые» каталоги помещаются ссылки («мягкие ссылки») на скрипты в /rc.d/init.d/, причём ссылки специальным образом именуются:

- первый символ K (kill) или S (start),

- следующие два символа — число в диапазоне [0..99] (порядковый номер), чтобы упорядочить запуск скриптов в нужной последовательности,

- имя скрипта.

При запуске система на последнем этапе конфигурирования заходит в каталог указанного уровня и запускает скрипты по порядку. Поскольку сначала по алфавиту идёт буква K, а потом S, то в этом порядке ссылки и выстраиваются: сначала всё убивается, а потом всё, что нужно для нормальной работы на данном уровне, снова запускается «вчистую». Получается, например, так:

. . .

K90network — 90-стая по порядку операция это «убиение сети»,

K92iptables — выключение fw,

K92messagebus — выключение «шины сообщений»,

S01sysstat — запись в протокол метки «LINUX RESTART»,

S02udevd — запуск демона udevd для формирования каталога /dev,

S05bridge — запуск моста между сетевыми интерфейсами,

. . .

S91nmd — запуск демона именования NetBIOS поверх именования IP,

S95alteratord — запуск демона для Центра управления системой

S99local — выдача приветствия системы «Welcome to hostname»: всё готово, входите.

Для облегчения работы администратора в начале стартовых скриптов помещается строчка

chkconfig: <список_уровней> <порядковый_номер S>
<порядковый_номер K>,

где в «списке уровней» перечисляются те номера «уровневых» каталогов, в которых ссылка для запуска сервиса должна быть сформирована, а порядковые номера — это те самые числа в диапазоне [0..99], определяющие порядковые номера скриптов, соответственно для старта сервиса и убиения сервиса.

Тем самым, администратору даётся подсказка, в каком порядке запускаются сервисы и в результате существенно снижаются требования к его квалификации — ему уже не надо вникать в суть сервисов и определять взаимозависимости, достаточно открыть скрипт в редакторе/ просмотрщике, прочитать эту строчку и создать в «уровневых» каталогах ссылки на стартовый скрипт с указанными номерами. Более того, в некоторых особо «умных» пакетах нужные ссылки создаются при инсталляции пакета автоматически.

### 7.1.4. Схема с суперсервером

Суперсервер — это сервис запуска сервисов по требованию (on demand). Он работает следующим образом:

- запускается с помощью стартового скрипта по схеме BSD или SystemV в зависимости от системы;

- считывает свой конфигурационный файл /etc/xinetd.conf, настраивается и переходит в режим демона, как описано выше;

- из конфигурационного каталога /etc/xinetd.d/ считывает все файлы, которые там есть; отбирает те, в которых значение переменной disable установлено в no, остальные файлы игнорирует; из выбранных файлов запоминаются значения переменных:

~ type — какой сервис — внутренний (обрабатывает сам xinetd) или внешний,

~ user — пользователь, от имени которого будет запущен сервис,

~ id — имя сервиса,

~ socket_type — тип сокета STREAM, DGRAM или RAW,

~ protocol — протокол, по которому работает данный сервис,

~ port — порт, который использует данный сервис,

~ server — абсолютный путь к исполняемому файлу демона сервиса и, возможно, другие параметры (см. man xinetd);

- всё запомнив, суперсервер слушает указанные порты и ждёт: когда в порт приходит пакет указанного протокола, то суперсервер запускает демон нужного сервиса, дальше обработку делает сам демон сервиса, кото-

рый завершив обработку и отправив ответ (или завершив сеанс с клиентом), завершается.

- и т. д.

Смысл использования суперсервера в том, что он позволяет не держать в памяти редко используемые сервисы и, тем самым, в некоторой степени разгрузить систему. Учитывая, что некоторые демоны создают дополнительные родственные процессы и/или потоки для взаимодействия с клиентом, то «разгрузка» системы может быть существенной.

Для того, чтобы для запуска какого-либо сервиса задействовать суперсервер, необходимо создать в каталоге /etc/xinetd.d/ текстовый файл (лучше с именем этого сервиса) определённого формата (см. man xinetd.conf — есть примеры), в котором указать значения переменных для этого сервиса и перезапустить суперсервер.

### 7.1.5. Схема systemd

Systemd – менеджер системы и сервисов (см. man systemd), обратно совместимый с ранее описанными схемами и предоставляющий такие полезные функции, как параллельный запуск системных сервисов во время загрузки, активацию демонов по требованию, поддержку срезов (снепшотов) состояния системы и логику управления сервисами, основанную на зависимостях. Иначе говоря, решает в одном месте все те вопросы, что ранее реализовывались разными схемами.

Схема systemd появилась и получила распространение уже в нашем веке в связи с появлением в массовых количествах многоядерных процессоров. В старых системах процесс как установки системы (копирование программ, обнаружение устройств, конфигурирование), так и просто очередной загрузки, шёл линейно, то есть, многоядерность, даже если она была, незадействовалась. С появлением многоядерных процессоров естественно возникло желание этот процесс ускорить за счёт распараллеливания работы.

Организует распараллеливание демон systemd, который, запускается как самый первый процесс (pid = 1) и действует как менеджер сервисов. Конфигурационные файлы демона находятся в каталоге /etc/systemd/, а стартовые скрипты запуска сервисов (которые здесь называются юниты — unit) располагаются в каталоге /lib/systemd/. Причём, стартовые скрипты могут организовыватся в группы («target»).

В скриптах указывается взаимозависимость сервисов («Requisite» — какие сервисы нужны данному, «After» — после чего грузить, «Before» —

что может грузиться после, «Conflicts» — с какими сервисами конфликтует за ресурсы, то есть, с какими не параллелить).

Соответственно, демон systemd, на основе этой информации, реализует непротиворечивое дерево запуска. Также в скриптах указываются протоколы, по которым работает сервис, сокеты и порты, которые слушает. Это позволяет демону systemd организовать запуск сервисов по требованию, аналогично сервису xinetd.

Пока эта схема запуска сервисов не является полной альтернативой ранее описанным схемам, то есть, она совместима с ними, работает прежде всего на ранних этапах загрузки системы и определяет состав и порядок запуска «системных» сервисов, то есть, обеспечивающих функционирование системы в целом. В конце процесса загрузки (при запуске «пользовательских» сервисов) всё равно используются вышеописанные схемы, в рамках которых пользователь может организовать запуск нужных уже ему сервисов. Тем не менее, тенденция такова, что схема systemd станет основной.

Для того, чтобы организовать запуск сервиса по схеме systemd необходимо:

а) создать файл запуска сервиса определённого формата (см. man systemd), например, мы разработали свой несложный сервис с демоном abrtd. Тогда создаём примерно такой файл:

```
[Unit]
Description=Daemon to detect crashing apps
After=syslog.target

[Service]
ExecStart=/usr/sbin/abrtd
Type=forking

[Install]
WantedBy=multi-user.target
```

В файле мы видим следующее:

- в секции [Unit] — описание и указание на то, что наш сервис может быть запущен только после того, как будет запущен сервис syslog;

- в секции [Service] — указываем путь к исполняемому файлу нашего сервиса и сообщаем демону systemd, как он узнает, что наш сервис успешно загрузился;

- в секции [Install] — сообщаем демону systemd, что наш сервис надо запускать после того, как будет выполнено все, что определено в цели multi-user.target.

б) созданный файл обзываем так: имя_сервиса.service (то есть, у нас это будет: abrtd.service) и копируем его в каталог /etc/systemd/system/.

в) затем даём команду на перезапуск демона systemd:

**# systemctl daemon-reload**

По этой команде демон systemd перечитает свои конфигурационные файлы, в том числе и нами созданный.

г) после этого можно запустить наш сервис командой

\# **systemctl start abrtd.service**

Останов сервиса:

**# systemctl stop abrtd.service**

Перезапуск сервиса:

**# systemctl restart abrtd.service**

Перезагрузка сервиса (сервис перечитывает свои конфигурационные файлы, не прерывая работы):

**# systemctl reload abrtd.service**

Плюсом этой схемы являются большая гибкость и большие функциональные возможности — systemd умеет не только сервисы запускать. Но есть и очень существенный минус: в настоящем виде эта схема намного сложнее в использовании любой из предыдущих и, соответственно, выше требования к квалификации администратора. То есть вернулись обратно почти на уровень схемы BSD, когда надо было знать правильный порядок загрузки сервисов. Вероятно, это изменится.

## 7.2. Применимость схем запуска

Схема BSD использовалась в старых версиях FreeBSD — отсюда и её название, а также используется в Linux в дистрибутивах slackware. Однако, в последних версиях наметился отказ от этой схемы, в силу сложности конфигурирования, и переход на схему systemV.

Схема systemV используется в дистрибутивах unix AIX, IRIX, HP-UX, SCO, UnixWare и других, а также в дистрибутивах Linux redhat/fedora, suse, altlinux, debian и других.

Схема с суперсервером используется во всех дистрибутивах.

Схема systemd начала использоваться в последние годы в целях ускорения процесса загрузки и конфигурирования посредством распараллеливания работы при запуске системы. Стимулом для применения этой схемы стало появление многоядерных процессоров.

В целом схемы BSD и systemV используются для запуска высоконагруженных сервисов, когда запросов от клиентов много, а в тех случаях, когда запросов мало, лучше использовать схему с суперсервером, которая позволяет экономить ресурсы системы.

## 7.3. Где здесь провайдер и что он делает?

### 7.3.1. Функции провайдера в части сопровождения сервисов

Провайдер выполняет следующие работы по сопровождению сервисов:

- создание и поддержка сервисов для клиентов (своих и не своих),
- обеспечение доступа своих клиентов к сервисам в Интернет (созданных другими провайдерами, организациями и просто клиентами чьими-то),
- обеспечение доступа из Интернет к сервисам своих клиентов.

### 7.3.2. Дополнительные функции провайдера

И кроме того, провайдер решает вопросы защиты

- прежде всего себя (своей инфраструктуры, включая свои сервисы),
- но может решать также вопросы защиты своих клиентов (по договору).

По договору, потому, что провайдер должен быть «прозрачен» для клиентов (своих и чужих), если иное не оговорено в статусе провайдера — например, законодательством страны. Пример: блокировка некоторых сайтов, сами знаете каких; в этом случае провайдер становится «непрозрачным».

## Вопросы «на засыпку»

1. Можно ли web-сервер запускать по схеме с суперсервеом?
2. Можно ли по схеме разового запуска запустить torrent?
3. Можно ли по схеме systemd запустить torrent?
4. что такое «уровневый» каталог?
5. Используется ли схема BSD на altlinux?

# Лекция 8. Стек TCP/IP. Примеры сервисов

## 8.1. Пример 1: почтовый сервис

### 8.1.1. Почтовый сервис — описание

Это один из самых «капризных» сервисов с очень высокими требованиями к правильной настройке сети. Для его работы необходимо, чтобы стек протоколов TCP/IP был сконфигурирован правильно, и прежде всего, правильно работало разрешение имён — резолвер и DNS: преобразование из символьной формы имени в цифровую и обратно (см. п. 5.8.).

Структура почтового сервиса показана на рисунке 41. Он состоит из следующих элементов:

- MUA (Mail User Agent) — почтовый клиент, например, Thunderberd;
- MTA (Mail Transport Agent) — почтовый транспортный агент, например, Postfix;
- MDA (Mail Delivery Agent) — почтовый агент доставки, например, procmail;
- MAA (Mail Access Agent) — агент доступа к почте, например, popper3d, или Dovecot.

Протоколы:

- SMTP (Simple Mail Transfer Protocol) — простой протокол передачи почты;
- POP3 (Post Office Protocol) — почтовый протокол версии 3;
- IMAP (Internet Message Access Protocol) — интернет протокол доступа к сообщениям.

Алгоритм обмена почтовым сообщением: Колян, сидя за своим компьютером, сочинил письмо своему другану Вовану, используя почтовый клиент (MUA) Thunderberd. Пусть почтовый адрес Вована — vovan@mv.ru. Когда Колян нажимает кнопку «Отправить», его Thunderberd формирует письмо (заголовок («конверт») и текст), соединяется с MTA почтового сервера провайдера Коляна по протоколу SMTP (порт 25) и передаёт ему письмо. MTA, получив письмо, обращается к сервису DNS, на предмет поиска записи MX (Mail Exchanger) для домена mv.ru, в котором, наверное, находится почтовый сервер адресата. Если такая запись найдена, то MTA Коляна связывается с MTA найденного почтового сервера, спрашивает, есть ли такой адресат и если есть, то передаёт ему письмо. Если Колян

ошибся адресом, то MTA Коляна сформирует письмо Коляну об ошибке, вкладывает в него письмо Коляна и отправит его Коляну, то есть, передаст своему MDA, который положит его в почтовый ящик Коляна (на hdd). Если же всё правильно, то MTA Вована передаст полученное письмо своему MDA, который положит его в почтовый ящик Вована (на hdd). Когда Вован надумает сесть за компьютер и почитать письма, он даст указание своему MUA «получить письма». Тогда MUA Вована соединится с почтовым сервером провайдера, конкретно, с MAA (например, dovecot, по протоколу POP3, порт 110, или по протоколу IMAP, порт 143), который отдаст ему письмо.

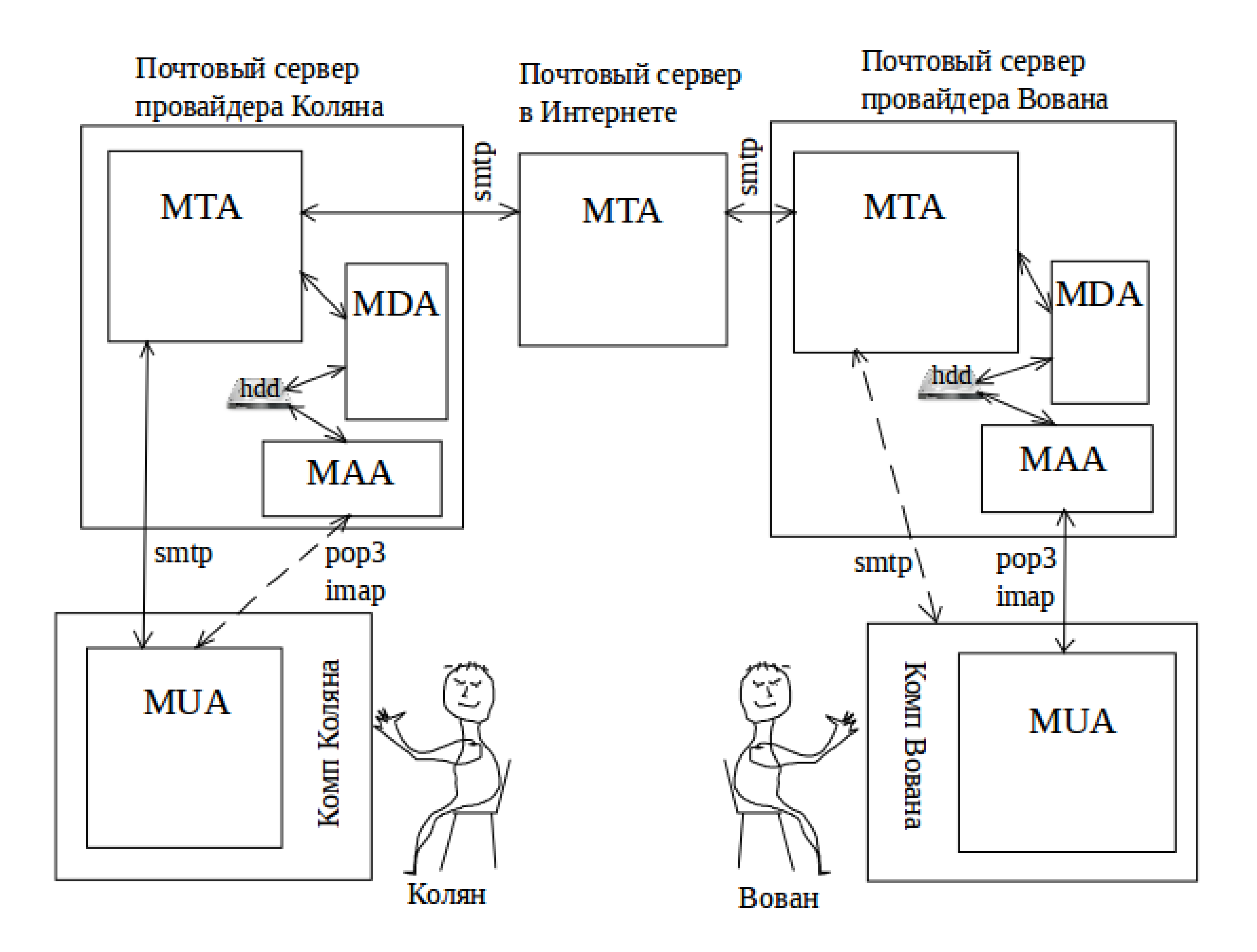



*Рис. 41.* Колян написал письмо Вовану

На рисунке пунктирами показаны взаимодействия:

- слева: Колян может получить письмо с сообщением об ошибке, если оно будет сформировано MTA;

- справа: Вован может послать ответ Коляну по вышеприведённому алгоритму.

### 8.1.2. Почтовый сервис — установка и настройка

Рассмотрим установку почтового сервиса на компьютере с установленной системой Альт Линукс 7.0.5 Кентавр. В качестве клиента может быть использован любой другой компьютер (даже Wind'овый) с установленным почтовым клиентом Thunderberd.

Порядок настройки:

**1. Компьютеры должны быть объединены в сеть физически,** то есть, в разъёмы RJ-45 (сетевые) должны быть вставлены кабели, которые должны подключаться к коммутатору.

**2. Должна быть настроена сеть,** как минимум так, как описано в приложении к заданию на лабораторную «Настройка сети без DNS» — это будет локальная сеть. Но вы можете аналогичным образом настроить и корпоративную сеть. Обязательно должна быть выполнена проверка правильности настройки сети, как описано в лабораторной: зайти на другой компьютер по краткому символическому имени другого компьютера и вернуться обратно в свой компьютер по краткому имени, а затем тоже самое повторить с полным именем. Хождение с компа на комп по IP-адресу не считается — на такой сети почтовый сервис работать не будет. Пусть мы настроили сеть так, как описано в приложении к лабораторной «Настройка сети без DNS».

**3. В качестве MTA будем использовать Postfix.** В Кентавре 7.0.5. он уже установлен по умолчанию, но настроен только на работу с интерфейсом lo0, то есть, для передачи писем только между локальными пользователями. Postfix использует для своей работы системных пользователей mail, mailman, postfix, postman и группы mail, mailman, postfix, postman, postdrop. В altlinux все нужные пользователи и группы создаются автоматически при установке пакетов. Также создаются стартовые скрипты запуска почтовой системы в /etc/rc.d/init.d/ и даже в уровневых каталогах присутствует правильная ссылка на запуск postfix и он реально запускается, но . . только для локальной работы.

**4. В качестве MDA в altlinux используется procmail,** он также уже установлен (вместе с Postfix) и настроен для раскладки приходящих писем по почтовым ящикам всех зарегистрированных в системе пользователей — локальных пользователей.

**5. Все конфигурационные файлы Postfix лежат в каталоге /etc/postfix/.** Для перенастройки Postfix для работы в сети необходимо поправить его основной конфигурационный файл main.cf следующим образом:

```
#/etc/postfix/main.cf
mailbox_command = /usr/bin/procmail — $DOMAIN —d $LOGNAME
inet_protocols = ipv4
inet_interfaces = all
myhostname = имя_компа.домен (например, mail.firma.ru —
        почтовый сервер фирмы)
mydomain = домен (например, firma.ru)
mynetworks = 192.168.0.0/16, 127.0.0.0/8
        (все сетки в диапазоне 192.168 и локальная петля)
myorigin = $myhostname (имя сервера, используемое в конверте)
mydestination = $myhostname, $mydomain, localhost.localdomain,
        localhost.$mydomain, \
mail.$mydomain (адресаты — локальные пользователи сервера и \
        пользователи нашей сети)
smtp_host_lookup = hosts (сеть без DNS, поэтому говорим,
        что разрешение имён — в hosts)
```

*Примечание.* Всё, что в круглых скобках, в конфигурационный файл вставлять не надо.

В переменной mynetworks перечисляются обслуживаемые сетки ip-адресов. Если почтовый сервер будет обслуживать только локальную сеть (например, 192.168.99.0), то её и указываем:

```
mynetworks = 192.168.99.0/24, 127.0.0.0/8
```

Поскольку у нас сеть без DNS, то переменной smtp_host_lookup назначаем значение hosts, чтобы Postfix не лез за разрешением имён к DNS (по умолчанию значение этой переменной — dns).

После исправлений проверяем правильность настроек:

```
# postfix check
```

и перезапускаем сервис:

```
# systemctl reload postfix
```

Тестируем — отправляем письма пользователям с помощью команды mail:

```
[user@mail ~]$ mail vasja
```

Subject: Proba
PROBA-точно_проба
Cc: user
Ctrl/D
[user@mail ~]$

Здесь наш компьютер — mail.firma.ru, наш логин — user. Пишем письмо пользователю vasja. В Cc: указываем, что хотим получить себе копию. Завершение письма — Ctrl-D с новой строки.

После отправки писем смотрим, что появилось в почтовых ящиках пользователей в каталоге /var/spool/mail/. Также нелишне заглянуть в протокол /var/log/maillog.

**6. Далее устанавливаем и настраиваем получение почты по протоколам POP3/IMAP**. Для этого с помощью synaptic ставим пакет dovecot, который по умолчанию не устанавливается. После установке в каталоге /etc/rc.d/init.d/ появится стартовый скрипт запуска данного сервиса. Создаём ссылки в каталоге /etc/rc.d/rc5.d/ для убиения и старта сервиса:

# ln —s ../init.d/dovecot K54dovecot
# ln —s ../init.d/dovecot S54dovecot

Исправляем конфигурационный файл /etc/dovecot/dovecot.conf

protocols = imap pop3 lmtp
listen = * (слушаем только интерфейсы ipv4)
verbose_proctitle = yes (увеличиваем количество информации о пользователях)

Запускаем сервис и проверяем с помощью команды ps:

[root@mail etc]# service dovecot start
[root@mail etc]# ps —ax | grep dovecot
2778 ?  Ss  0:00 /usr/sbin/dovecot —F
2781 ?  S  0:00 dovecot/anvil [0 connections]
2782 ?  S  0:00 dovecot/log
2783 ?  S  0:00 dovecot/ssl-params
2784 ?  S  0:00 dovecot/config
2785 ?  RN  0:02 dovecot/ssl-params
2787 pts/0 S+  0:00 grep —-color=auto dovecot
[root@mail etc]#

**7. Далее настраиваем клиенты для отправки/получения почты**. Как уже было сказано, на компьютерах локальной/корпоративной сети у нас предполагается использовать MUA Thunderberd. Соответственно, в настройках Thunderberd указываем в качестве имени почтового ящика свои логин в формате login@hostname_почтового_сервера и пароль, с которыми мы зарегистрированы на почтовом сервере, то есть, в имени почтового ящика мы указываем свой логин на почтовом сервере и его (почтового сервера) полное имя в локальной/корпоративной сети. Нажимаем OK. При этом Thunderbird проверяет правильность пары логин-пароль на указанном компьютере сети — почтовом сервере и если на этом компьютере есть такой пользователь с таким паролем и этот компьютер является почтовым сервером, то Thunderbird сообщает: «Конфигурация найдена у провайдера электронной почты» — см. рис. 42. Нажимаем «Готово». Если вам есть письма, то Thunderbird их вам скачает.

Если у вас несколько почтовых ящиков, то далее их можно добавить.

Теперь каждый раз при запуске Thunderbird, он будет автоматически проверять наличие писем вам на почтовом сервере, а также каждый раз при отправке вами письма. То есть, при каждом подключении в почтовому серверу Thunderbird автоматически выполняет полный цикл работ по приёму/отправке писем.

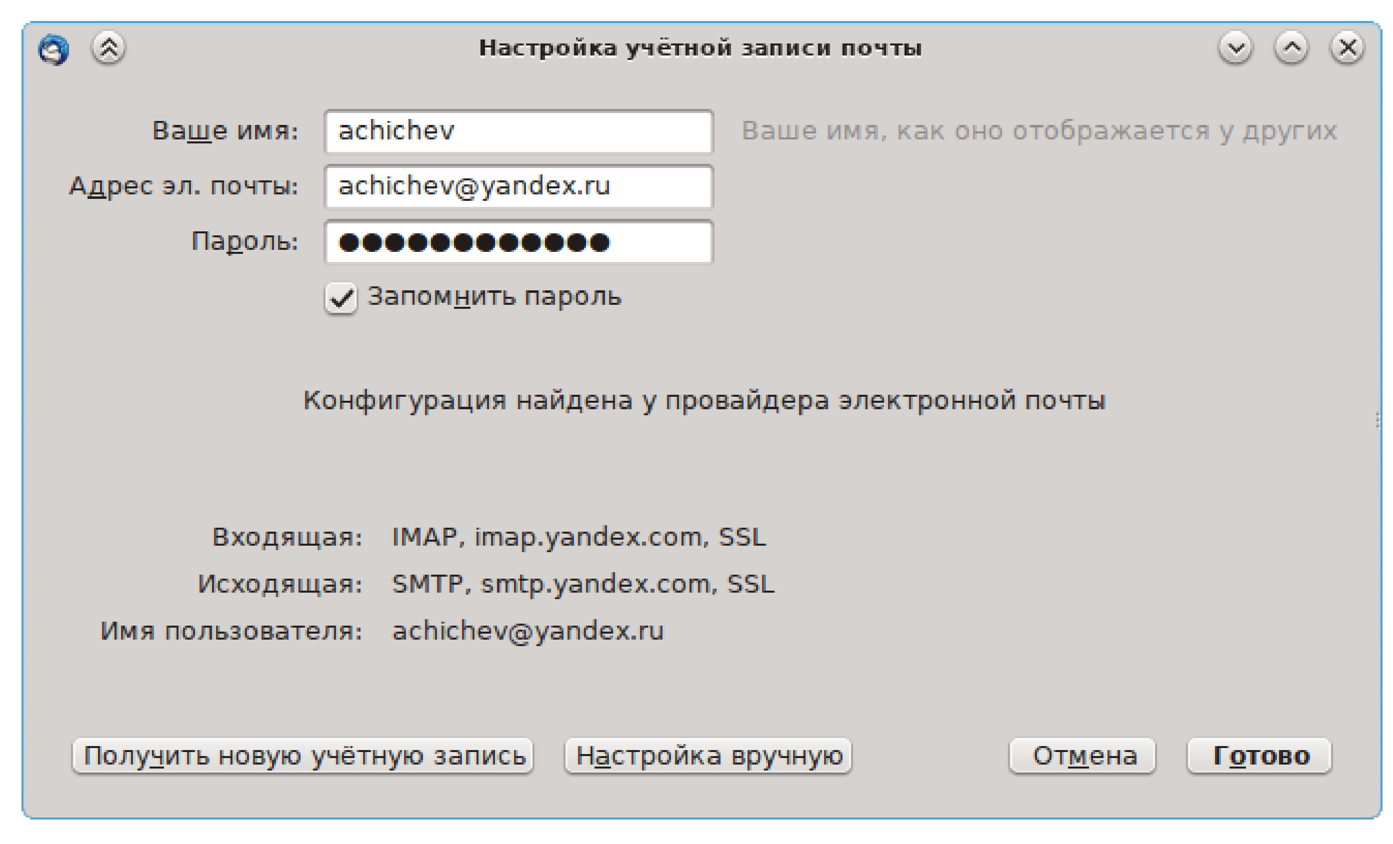



*Рис. 42.* Настройка Thunderbird

## 8.2. Пример 2: сервис ftp

В качестве примера рассмотрим установку и настройку сервера wsftp. Снова установка пакета wsftp с помощью synaptic из репозитария altlinux. Напоминаю: если хотите всё установить легко и быстро, то не следует пользоваться чужими пакетами, всегда надо предпочитать пакеты из родного дистрибутива.

Сервис ftp в отличие от многих других «узкопрофессиональных» сервисов является многофункциональным. Он реализует следующие фуекции:

- доступ «по чтению» к некоторому расшариваемому каталогу (обычно pub) с файлами;
- доступ в некоторый каталог (по умолчанию — incoming) с возможностью записи в него своих файлов для обмена с другими пользователями сервиса или просто для хранения;
- если пользователь зарегистрированный в системе, то может предоставляться доступ (чтение и запись) в домашний каталог пользователя;
- возможность переброса файлов с одного ftp-сервера напрямую на другой ftp-сервер.

Доступность функций определяется настройками сервиса и они могут предоставляться в различных сочетаниях независимо друг от друга.

Конфигурационный файл wsftp по умолчанию находится в файле /etc/vsftpd.conf

**Пример настройки** сервиса ftp с доступом только для зарегистрированных пользователей в каталоги pub — чтение и incoming — чтение и запись. В файле vsftpd.conf следует изменить следующие строчки:

```
listen=YES
anonymous_enable=NO
local_enable=YES
write_enable=YES
chroot_local_user=YES
```

Остальные настройки — по умолчанию.

Рабочий каталог сервиса ftp обычно создаётся в /var/ftp. В этом каталоге нужно создать два подкаталога pub и incoming. Права на каталог pub — 755, а на каталог incoming — 777. На каталог incoming необходимо также установить sticky-бит.

При установке пакета wsftp создаётся системный пользователь ftp. Этот пользователь должен быть назначен хозяином каталога /var/ftp и всех его подкаталогов.

Если мы хотим, чтобы ftp-сервисом пользовались не только зарегистрированные пользователи, но и некоторые другие, кого мы известим, но не anonymous!, тогда создаём вспомогательного обычного пользователя, например, ftpuser с домашним каталогом /var/ftp и определяем ему пароль.

О логине-пароле извещаем определённую группу пользователей.

## 8.3. Пример 3: сервис nfs (network file system — сетевая файловая система)

### 8.3.1. Назначение и смысл сервиса

Это весьма сложный сервис, ещё сложнее, чем почтовый, который (O_o) очень легко настраивается. В основных дистрибутивах altlinux (Workstation, Kworkstation, Simply, Образование) он уже предустановлен, но не настроен.

Однако, также как и почтовый сервис, nfs весьма требователен к точной настройке сети.

Компьютер с запущенным и настроенным сервисом nfs будет предоставлять доступ внешним пользователям к некоторым каталогам своей файловой системы, то есть, будет «расшаривать» в сеть некоторые свои каталоги. Такое предоставление доступа называется "экспортом": говорят, что система экспортирует свои каталоги. Как именно и кому каталоги будут экспортироваться, определяется списком, который задает системный администратор. В системах linux этот список содержится в файле /etc/exports — именно редактированием этого файла и настраивается nfs.

### 8.3.2. RPC (Remote Procedure Call — удалённый вызов процедур)

Сервис nfs работает посредством механизма удаленного вызова процедур (RPC – Remote Procedure Call).

Идеология RPC очень проста и привлекательна для программиста. Как обычно работает сетевое приложение? Оно следует некоему протоколу (например, HTTP): формирует пакет с запросом, вызывает системную функцию установления соединения, затем функцию отправки пакета, затем ждет ответного пакета и вызывает функцию закрытия соединения. Это значит, что вся работа с сетью является заботой программиста, который пишет приложение: он должен помнить о вызове функций сетевого API системы, думать о действиях в случае сбоев сети.

RPC предполагает иной способ обмена данными между клиентом и сервером. С точки зрения программиста, приложение клиента, работающее с помощью RPC, вызывает функцию на сервере, она выполняется и возвращает результат. Пересылка запроса на выполнение функции через сеть и возврат результатов от сервера клиенту происходит незаметно для приложения, поэтому последнее не должно беспокоиться ни о сбоях сети, ни о деталях реализации транспортного протокола.

Для того, чтобы обеспечить прозрачность пересылки данных через сеть, придумана двухступенчатая процедура. На сервере любое приложение, которое хочет предоставлять свой сервис через RPC, регистрируется в программе, которая называется транслятором портов (port mapper). Функция этой программы – устанавливать соответствие между номером процедуры RPC, которую запросил клиент, и номером TCP или UDP порта, на котором приложение сервера ждет запросы. Вообще говоря, RPC может работать не только с TCP или UDP, например, в Solaris работа RPC реализована на базе механизма TI (Transport-Independent), поэтому в Solaris транслятор портов называется rpcbind, а не portmap, как в Linux или FreeBSD.

Приложение, которое регистрируется у транслятора портов, сообщает ему номер программы, номер версии и номера процедур, которые могут обрабатываться данной программой. Эти процедуры будут впоследствии вызываться клиентом по номеру. Кроме этого, приложение сообщает номера портов TCP и UDP, которые будут использоваться для приема запросов на выполнение процедур.

Клиент, желающий вызвать выполнение процедуры на сервер, сначала отправляет запрос транслятору портов на сервер, чтобы узнать, на какой TCP или UDP порт надо отправить запрос. Транслятор портов запускается при старте системы и всегда работает на стандартном порте 111 (см. файл /etc/services). Получив ответ от него, клиент отправляет запрос на тот порт, который соответствует требуемому приложению. Например, сервер NFS работает на порту 2049.

### 8.3.3. Процедура монтирования общего каталога через NFS

Клиент NFS посылает запрос на монтирование удаленному компьютеру, который предоставляет свою файловую систему (обычно – некоторую ее часть) для этого пользователя. При этом говорят, что сервер NFS "экспортирует" тот или иной каталог (подразумевается – с подкаталогами). Запрос от клиента NFS попадает на обработку демону mountd. Тот выдает

клиенту NFS специальный ключ. Этот ключ является идентификатором, который однозначно идентифицирует каталог, смонтированный по сети.

По NFS можно смонтировать как целые файловые системы, так и отдельные каталоги. Из соображений безопасности запрещено монтировать каталоги "через раздел". Это означает, что если каталог /var расположен на одном разделе диска, а каталог /var/adm – на другом, то при монтировании каталога /var каталог /var/adm не будет автоматически смонтирован. Если требуется монтировать те подкаталоги экспортируемого каталога, которые расположены в другой файловой системе (на другом разделе), следует экспортировать их отдельно и указывать в /etc/exports еще один разделяемый каталог – тот самый подкаталог с другого раздела.

Ключ, выданный клиенту при монтировании и идентифицирующий сеанс работы с данным удаленным каталогом, сохраняется при перезагрузке NFS-сервера, чтобы после восстановления его работы клиенты, замершие в ожидании, продолжили работу с удаленным сервером как ни в чем ни бывало. При демонтировании и новом монтировании файловой системы через NFS ключ обычно изменяется. На практике перезагрузка NFS-сервера все-таки может привести к сбою в работе клиентского приложения, особенно, если приложение читает или записывает файл в удаленный каталог во время перезагрузки. Например, если программер не умеет обрабатывать таймауты или вызывает функции без проверки кода возврата.

После монтирования файловой системы через NFS клиент посылает серверу запросы на передачу и прием файлов, эти запросы обрабатывает демон nfsd.

Демонтирование файловой системы выполняется также, как и демонтирование любой другой файловой системы – командой umount.

Ниже будут обсуждены следующие аспекты настройки службы клиент-сервер в сети:

- определение списка каталогов на сервере, которые должны быть общими;
- определение прав доступа к этим каталогам;
- аутентификация и назначение прав доступа в NFS;
- настройка сервера NFS, запуск необходимых программ;
- настройка клиентов NFS, монтирование удаленных каталогов.

### 8.3.4. Настройка NFS-сервера

Для настройки NFS сервера требуется выполнить настройку как минимум трех приложений: portmap, mountd и nfsd. Описание ниже касается версий 2 и 3 сервиса nfs, более современная версия 4 несколько отличается.

#### 8.3.4.1. portmap: объявление служб RPC

Для начала следует запустить программу portmap, если она еще не запущена. Скорее всего, она запускается при старте вашей системы altlinux. Эта программа преобразует номера вызовов процедур RPC в номера портов TCP и UDP. При запуске любого RPC-сервера, т.е. программы, работающей с протоколом RPC, программа portmap получает от этого RPC-сервера информацию о том, какие номера процедур RPC он намерен обслуживать и через какой порт TCP (UDP) ему следует направлять запросы.

Когда клиент деалет RPC-вызов, сначала происходит выяснение требуемого номера порта на машине сервера у portmap.

Поэтому portmap должна быть запущена до того, как будет запущен любой из RPC-серверов. При аварийном завершении portmap необходимо вначале перезапустить portmap, и затем перезапустить все RPC-серверы.

Для проверки готовности всех служб NFS к работе через portmap используется команда rpcinfo —p:

#### 8.3.4.2. Запуск сервиса экспорта файловых систем

Сервис NFS предоставляется двумя программами, которые обрабатывают соответствующие RPC-запросы. Это программы mountd и nfsd.

Программа mountd обрабатывает запросы на удаленное монтирование файловых систем. Для получения списка экспортируемых каталогов применяют команду showmount –e, которая обращается к mountd за информацией:

showmount —e
export list for sunny:
/nfst (everyone)

Для того, чтобы на сервере NFS узнать, какие системы подсоединили к себе разделяемые каталоги этого сервера, следует дать команду showmount без параметров:

showmount
www.eu.spb.ru

Программа nfsd – это обработчик файлового запроса удаленного клиента NFS к файловой системе сервера NFS.

Параметры сервиса NFS настраиваются в файле /etc/default/nfs, а запуск и остановка сервиса осуществляется соответственно командами

/etc/init.d/nfs.server start

и

/etc/init.d/nfs.server stop

В файле /etc/default/nfs следует указать достаточное количество потоков, которые можно параллельно запускать для обслуживания одновременных запросов к файлам через NFS. За это отвечает параметр NFSD_SERVERS.

Значение этого параметра не должно быть меньшим максимального количества одновременных обращений к разделяемым файловым системам NFS на сервере. Слишком маленькое число потоков вызовет задержки в работе клиентов, поскольку им придется становиться в очередь на обработку запросов. В то же время, излишне большое число приведет к нерациональному расходу памяти NFS-сервера. Оптимальное число определяется из опыта, и единственное, что можно сказать определенно: оно не должно быть больше удвоенного общего количества компьютеров сети, которые настроены для работы через NFS. По умолчанию это число равно 16, что для нагруженных серверов NFS очевидно мало.

Если из-за слишком большого количества потоков nfsd загрузка процессора возрастет до 100%, имеет смысл уменьшить число этих потоков. С одной стороны, идеально иметь 2 потока на каждого активного клиента, постоянно обращающегося к ресурсам NFS, с другой – на один процессор рекомендуется запускать не более 16 потоков, если это достаточно медленный процессор одноядерник.

Перед запуском mountd и nfsd следует убедиться, что в /etc/exports указаны все каталоги, которые вы собираетесь экспортировать, а также разумно настроены параметры безопасности.

#### 8.3.4.3. Аутентичность пользователей NFS

Работа с общим диском, разделяемым между многими пользователями сети, предполагает, что в пределах сети существует общее пространство имен пользователей, т.е. пользователь vasja на любом клиентском (в терминах NFS) компьютере имеет то же реальное имя и (что важнее) тот же идентификатор (UID), что и пользователь vasja на сервере NFS. Это достигается использованием централизованной аутентификации (например, с помощью PAM и сервера аутентификации или с помощью NIS+). Кроме этого, важно ограничить права пользователя root при доступе через NFS, т.к. пользователь root на любом компьютере в любой системе UNIX имеет идентификатор 0, но от имени пользователя root на разных компьютерах могут работать разные люди.

Для ограничения доступа пользователя root на сервере NFS любые файловые запросы от имени пользователей root клиентских компьютеров выполняются от имени nobody. Можно указать, пользователям root каких

компьютеров мы предоставляем привилегированный доступ от имени root и через NFS.

По умолчанию файловая система экспортируется с правами чтения и записи для тех, кто ее смонтирует. Однако, права доступа к конкретным каталогам могут запрещать запись в них; фактически права доступа к удаленной файловой системе определяются комбинацией прав, данными при монтировании системы и прав доступа к каталогам [3]; силу имеют более строгие права (например, нельзя записать файл в каталог, если нет права записи в каталог или если право есть, но файловая система экспортируется в режиме read-only – только для чтения).

В простейшем случае, файл /etc/exports является единственным файлом, требующим редактирования для настройки NFS-сервера. Данный файл управляет следующими аспектами:

- какие клиенты могут обращаться к файлам на сервере,
- к каким иерархиям каталогов на сервере может обращаться каждый клиент,
- как пользовательские имена клиентов будут отображаться на локальные имена пользователей.

Каждая строка файла exports имеет следующий формат:

точка_экспорта клиент1(опции) [клиент2(опции) …],

где:

точка_экспорта — абсолютный путь экспортируемой иерархии каталогов,
клиентN — имя одного или более клиентов или IP-адресов, разделенные пробелами, которым разрешено монтировать точку_экспорта. Опции описывают правила монтирования для клиента, указанного перед опциями.

Пример файла exports:

/archiv1 files(rw,sync) 10.0.0.1(ro,sync) 10.0.230.1/24(ro,sync)

В данном примере компьютерам files и 10.0.0.1 разрешен доступ к точке экспорта /archiv1, при этом, хосту files на чтение/запись, а для хоста 10.0.0.1 и подсети 10.0.230.1/24 доступ только на чтение.

Описания хостов в /etc/exports допускается в следующем формате:

- имена отдельных узлов описываются, как files или files.DOMAIN.local.
- описание маски доменов производится в следующем формате: *DOMAIN.local включает все узлы домена DOMAIN.local.

- подсети задаются в виде пар адрес IP/маска. Например: 10.0.0.0/255.255.255.0 включает все узлы, адреса которых начинаются с 10.0.0.

- задание имени сетевой группы @myclients, имеющей доступ к ресурсу (при использовании сервера NIS).

Полный список доступных режимов и настроек при указании экспортируемой файловой системы доступен в man share и man share_nfs.

### 8.3.5. Дополнительные возможности сервиса

Помимо основной функции (расшаривание каталогов в сеть) — сервис nfs может использоваться для реализации дополнительных функций:

- создание бездисковых рабочих станций;
- блокировка файла (в новых версиях nfs — частей файла) на nfs-сервере для обеспечения групповой работы с одним и тем же файлом;
- поддержка дисковых квот для пользователей;
- кеширование файловой системы носителя, если носитель с низкой скоростью доступа (например, CD-ROM или флешка).

### 8.3.6. Настройка клиента. Монтирование удаленных файловых систем

Монтирование файловых систем NFS осуществляется очень похоже на монтирование файловой системы любого другого типа. Пример:

```
# mount -t nfs 10.0.230.2:/archiv1 ./nfst
```

Здесь:

-t nfs — указывает, что монтируется файловая система по nfs,

10.0.230.2 — адрес сервера nfs, где расположен монтируемый (расшариваемый) каталог archiv1,

./nfst — точка монтирования.

В чём преимущество сервиса nfs перед другими способами «расшаривания» каталогов? В том, что файлы оказываются доступными для обработки так, как будто они находятся на локальном носителе, а вовсе не на удалённом. То есть, пользователь может их редактировать и даже запускать на исполнение — другие способы «расшаривания» файлов таких возможностей не предоставляют.

## 8.4. Пример 4: сервис www

В старину, когда деревья были еще маленькие, а ЭВМ ещё большие, www сервер (это был, конечно, apache ещё 1-ой версии) настраивался очень просто: у него был один конфигурационный файл httpd.conf, в котором нужно было изменить пару строк. И всё. После перезагрузки системы мы получали работающий web-сервер, который нужно было только заполнить контентом.

Увы, теперь деревья уже большие, а (почти все) ЭВМ сосем маленькие и apache, теперь уже 2-ой версии, настраивается существенно сложнее. У него теперь целый каталог конфигурационных файлов /*etc/*httpd2/ и очень много возможностей с помощью этих конфигов настраиваемых.

Тем не менее, как настроить apache быстро и легко? Предположим, что перед нами задача: быстро получить работающий web-сервер, например, для сдачи лабы.

Для этого сделаем следующее:

1. В файле /etc/httpd2/conf/sites-available/default.conf исправим параметр RewriteCond. Заменим

%{HTTPS} != on

на

%{HTTPS} != off.

2. В файле /etc/httpd2/conf/sites-available/default.conf находим строку:

DocumentRoot "/usr/share/doc/indexhtml/ "

и заменяем её на свою

DocumentRoot "/var/www/html/"

то есть, мы будем создавать свой контент в каталоге /var/www/html/, в частности, головной файл index.html нашего сайта будет лежать по пути /var/www/html/index.html

В этом же файле находим строку:

<Directory "/usr/share/doc/indexhtml/">

и заменяем её на

<Directory "/var/www/html/">

3. В файле /etc/httpd2/conf/include/Directory_html_default.conf в строку

Options Includes FollowSymLinks MultiViews

допишем слово Indexes, то есть, строка будет выглядеть так:

Options Indexes Includes FollowSymLinks MultiViews

4. Перезапускаем apache:
#service httpd2 restart

и заходим на сервер браузером по адресам:
http://ip_адрес_нашего_компа

и

http://localhost
ip-адрес нашего компа можно увидеть командой ifconfig.

Всё. Осталось создать контент. Для этого смотрите задание на лабораторную работу — что там нужно создать?

Сервис www в настоящее время, пожалуй, самый развитой из всех используемых сервисов. То есть, очень часто он используется не только для решения основной своей задачи — выдачи html-страниц, пусть даже и динамических, но и как основа для создания мультисервисных систем. Это делается посредством усложнения модуля «бизнес-логика».

Но в данном пособии этот вопрос не рассматривается.

## 8.5. Пример 5: сервис torrent

Самый простой способ запустить на компе сервер torrent это воспользоваться пакетами transmition или deluge. Оба пакета имеют клиент-серверную архитектуру, то есть, содержат в своём составе и клиент torrent и сервер. Настройки пакетов — из меню клиента, но можно и непосредственно править конфигурационные файлы. Например, ~/.config/transmission

## 8.6. Пример 6: сервис «социальная сеть»

Для запуска сервиса типа «социальная сеть» можно

- воспользоваться готовыми движками (скриптами) для создания социальной сети, (например, на PHP) со стандартными для подобных продуктов функциями, чатами, микроблогами а-ля Twitter и другими популярными функциями (вводим в браузере: «скачать движок социальной сети»),
- либа написать свой движок (скрипт) для создания социальной сети на каком-либо языке (на PHP, Perl, java или даже на Си).

Очевидно, что этот сервис создаётся поверх web-сервиса. То есть, нужно уметь запускать и конфигурить apache. И этот движок (скрипт) — это та самая бизнес-логика архитектуры SOA.

## 8.7. Заключительные замечания

Сервисы — один из самых сложных видов ПО. Их повышенная сложность обуславливается тем, что это сетевое ПО. То есть, задействуются не только то ПО, что установлено в компе, но и то, что установлено в соседнем компе, и в компе за соседним компом, и вообще, в где_то_там_компе.

Отсюда очевидный вывод: сервисы требуют серьёзного отношения. К установке и настройке. Неоднократно приходится наблюдать, как будущие_админы пытаются что-то настроить на компе, в котором уже кто-то что-то делал, конфигурил. Что он там делал — чаще всего неизвестно, а иногда и известно, но это мало поможет. И вот будущий_админ «корячится», даже сильно старается, но ничего не выходит. Почему? Ответ очевиден: система приведена в некоторое состояние, в котором настраиваемый сервис работать не будет, что-то в ней сделано не так, что-то включено лишнее, мешающее, что-то отключено нужное.

1. Следовательно: хотите легко и быстро настроить сервис — вы должны быть уверены, что система «чистая». Например, свежеустановленная. Тогда, в большинстве случаев проблем не будет и сервис будет настраиваться «по инструкции».

2. Некоторые сервисы взаимозависимы, то есть, требуют предварительной установки и настройки других сервисов. Об этом часто забывают.

3. Сервисы имеют архитектуру «клиент-сервис», то есть, нужно устанавливать и настраивать все компонены сервиса: сервер, клиент, протокол, бизнес-логика, СУБД. Об этом тоже часто забывают.

4. Некоторые сервисы несовместимы: либо тот, либо другой. И если вы не знаете, что ставилось на компе — возможно будут проблемы. Более того, нередко просто снос мешающего сервиса не достаточен. Ибо сервис — это комплекс ПО, именно комплекс. В состав которого нередко входит не только ПО самого сервиса (см. выше пункт 3), но и что-то ещё: библиотеки, модули ядра, конфигурационные файлы и прочее. И это «что-то_ещё», установленное при установке сервиса, как правило, не удаляется из системы при сносе самого сервиса, а остаётся в системе и может конфликтовать с новым устанавливаемым сервисом.

5. Поскольку сервис — это комплекс ПО, то вполне возможна ситуация, когда компоненты сервиса разных версий могут не заработать друг с другом, или заработают, но частично потеряют функциональность.

6. Крайне желательно ставить сервис с того с того же дистрибутива, с которого установлена система, с репозитария дистрибутива, точнее, с репо-

зитария версии дистрибутива, если сервис есть в репозитарии. Если в репозитарии данного сервиса нет и его приходится ставить с иного источника, например, с сайта разработчиков сервиса, то по крайней мере, ознакомьтесь с рекомендациями разработчика дистрибутива по установке данного сервиса в документации, на форуме дистрибутива или ином источнике, поддерживаемом разработчиками дистрибутива для своих пользователей.

Поэтому, должно действовать «железное правило»: СЕРВИС ВСЕГДА ДОЛЖЕН СТАВИТЬСЯ НА ЧИСТУЮ СИСТЕМУ. И самое простое здесь — на свежеустановленную. Если же такого состояния невозможно добиться (например, необходимо поставить дополнительный сервис на работающую систему), то, увы вам, придётся разбираться и может быть долго.

**Вопросы «на засыпку»**

1. Как передаются изображения (фото, например) по e-mail?
2. Является ли «в контакте» сервисом?
3. Что такое конверт письма и что на нём пишется?
4. Что такое RPC?

5*. Почему к некоторым серверам www нужно обращаться так: www.поддомен.домен.ru, а к некоторым можно просто поддомен.домен.ru?

# Лабораторные работы

## ЛАБОРАТОРНАЯ РАБОТА № 1

**Тема:** Сетевые сервисы. Запуск web-сервера apache. Создание сайта-портфолио.

**Рекомендации и требования**

**Пункты 1, 2 и 3 выполнять студентам 2-го курса и старше.**

**0. Прежде чем начнёте — рекомендуется обновить систему:**

**- либо нажав APT-индикатор в трее: откроется окно обновления, нажать «автоматическое обновление», включится Synaptic, согласиться с его предложениями по обновлению; в конце процесса появится окошко, в котором будет написано «Done» («Сделано»), нажать Enter — обновление системы сделано;**

**- либо обновление можно сделать в терминале от root'a: даём команды**

**apt-get update; apt-get upgrade; apt-get dist-upgrade <Enter>** одной строкой и ждём завершения процесса обновления.

**1.** Необходимые пакеты рекомендуется устанавливать с помощью synaptic. В качестве репозитария использовать штатный диск дистрибутива, с которого ставилась система, либо штатный репозитарий дистрибутива в Интернете.

Ниже в Приложении 1 к лабораторной работе сказано, как настроить synaptic на компах лаборатории.

**Примечание 1 для «свежесрезанных» и тех, кто «в танке»:** в Приложении 1 сказано, как настроить synaptic на компах лаборатории.

**Примечание 2 для «свежесрезанных» и тех, кто «в танке»:** команды ubuntu не работают просто так на стандартных дистрибутивах Linux, например, на ALTLinux. Пользуйтесь документацией ALTLinux! — http://altlinux.org

**2.** Обеспечить запуск web-сервера при старте ПЭВМ. Как это сделать — смотреть в «Руководстве администратора AltLinux» — не обязательный пункт.

**3.** Web-сервер должен быть виден в локальной сети lab326.

**Первый курс начинает отсюда.**

**4.** Создать контент:

**- через меню главной странички обеспечить доступность отчётов по всем ранее выполненным лабам в формате html, причём, каждый отчёт на отдельной страничке (в отдельном файле).**

В приложенном к заданию каталоге shablon-site имеется пример очень простого шаблона, который можно использовать — файл index.html — загляните внутрь этого файла (в html-текст). Этот файл в кодировке cp-1251, то есть, в виндовой. Если измените кодировку, то не забудьте изменить и указание о ней, то есть, в строчке:

<meta http-equiv="content-type" content="text/html;charset=windows-1251">

укажите свою кодировку.

**5.** Можете добавить на сайт что-то ещё.

**6.** Найти бесплатный «лёгкий» хостинг и создать сайт на нём. Хостинг «лёгкий», если пустая страница (пустой шаблон) «весит» не более 20-30 килобайт. **Антипример:** пустая страница на ucoz.ru весит полмегабайта (!), поэтому если вы хотите этим хостингом пользоваться, то создайте свой шаблон и ни в коем случае не используйте готовый.

То есть, шаблон создавайте себе сами: то очень важное умение — создание шаблона под некоторые похожие заказы. На этом люди деньги делают достаточные, чтобы прожить в Ульяновске.

**7. Мероприятия по раскрутке сайта.**

**7.1. Корректность.** Роботы учитывают правильность оформления страничек сайта (по стандарту HTML)— см. руководства в каталоге shablon-site. Если странички оформлены неправильно, то сайту снижается рейтинг (сайт «банится»). При неправильном оформлении другие мероприятия по раскрутке приносят только временный успех — сайт всё равно в конце концов «уезжает» в конец рейтинга.

**7.2. Взаимозависимость-1.** Создать ссылки на сайты других студентов группы и/или друзей/знакомых. Попросите их, чтобы они сделали на своих сайтах ссылки на ваш сайт. То есть, сайт не должен быть сам по себе, он должен быть **в системе сайтов.**

**7.3. Взаимозависимость-2.** Сделать ссылки на сайты с документацией по темам лабораторных и лекций в Интернете. Очень хорошо делать ссылки непосредственно из текста отчёта по лабе (как в википедии).

**7.4. Контент.** Создать на сайте раздел «Библиотека» (или что-то аналогичное) и выложить в этом разделе документацию (например, учебники,

техническую документацию, художественную литературу или ещё что-то «творческо-необычное», что может заинтересовать посетителей сайта).

**7.5. Мобильность.** Страницы сайта должны быть «динамическими», то есть, изменяться в соответствии с окном браузера. Чтобы можно было и удобно было ходить на сайт с мобильных устройств.

**7.6. Поставить счётчики посещений сайта.** Особенно желателен счётчик, учитывающий страну, ОС, браузер, посмотренные страницы сайта. Совсем будет хорошо, если вы такой счётчик напишите самостоятельно и встроите в сайт.

**7.7. Добейтесь того, чтобы ваш сайт индексировался yandex'ом.** Найдите ваш сайт в поиске yandex'а по ключевым словам или фразам, то есть, по содержанию контента.

**7.8.** Помните: хороший сайт — это сайт с нужными материалами (нужной людям информацией!) и, в тоже время, в меру лёгкий («быстрый»). То есть, не следует делать сайт слишком «мультимедийным». «Мультимедийный» сайт — это сайт на любителя: круто, но иногда бесполезно; если канал слабый, то на него не зайдёшь.

**8. Мероприятия по поддержке сайта,** чтобы сайт существовал долго.

**8.1. Крайне желательно иметь на сайте оригинальный контент**, то есть, вами лично написанные статьи. Сочинения в школе писали. Вот и пишите для своего сайта. Информации в Инете, о чём писать, более чем достаточно. Роботы при сканировании Инета для всех копий стараются найти оригинал — тот первоисточник, откуда «содрали». Найдя, всем копиям снижают рейтинг. И помните: работа инженера — это работа с документацией.

**8.2. Объём сайта** должен быть достаточно большой, и чем больше — тем лучше для поддержки сайта, больше вероятность, что материал кому-то понадобится.

**8.3. На сайт нужно регулярно заходить**, не менее раза в неделю, и что-нибудь на нём править: добавлять, изменять и т. д. То есть, сайт должен быть постоянно «в работе». Иначе некоторые хостинги считают, что сайт «брошен» и удаляют его.

**8.4. Пункты 8.1, 8.2, 8.3 — пожалуй самые важные.**

**Раскрученный сайт — ваше «лицо».** Он утверждает вас как специалиста и доказывает, что вы реальный специалист. Пойдёте устраиваться на работу, скажете: «А вот у меня сайт есть. Он, конечно, на бесплатном хостинге. Но! Он существует уже несколько лет, он ищется yandex'ом и на него постоянно ходят!».

**Эти слова:**

**-** *на бесплатном* - да, я знаю цену денег!;

- *существует* - денег не плачу, а он существует!

- *ищется* - да, я умею раскручивать!

- *ходят* - да, я знаю, как заинтересовать людей!

Эти слова определяют вашу крутость как специалиста — вы не зря в университете учились, вы научились. По крайней мере, в сайтах вы что-то понимаете.

**Порядок сдачи лабораторной**

В отчёте о выполнении данной лабы должны быть:

- задание на лабу;
- описание порядка запуска и настройки web-сервера (apache);
- screen окна xterm с выполненной командой ps —ax, показывающей, что apache запущен;
- screen окна firefox, показывающий главную страничку своего сайта;
- **screen окна firefox, показывающий страничку поиска yandex'a с вашим сайтом;**
- продемонстрировать доступность web-сервера в лаборатории:
- локальной версии, установленной на компе лаборатории,
- версии в Интернете с отчётами по выполненным лабораторным работам.

**Эту лабораторную сдать также в электронном виде архивом**

**Срок сдачи лабораторной** — до ДД.ММ.ГГ.

**ПРИЛОЖЕНИЕ 1 к лабораторной №1**

**1. Настройка synaptic (программы управления пакетами)**

1.1. Программа **synaptic** доступна только пользователю root — администратору системы. Почему так, надеюсь, понятно.

Поэтому при запуске программы потребуется ввести пароль root'a.

1.2. Кроме того, для работы программы synaptic требуется доступность Интернета. В университете настройки сети для доступа в Интернет получаются автоматически по протоколу DHCP.

То есть, «Центр управления системой» → «Сеть» → «Ethernet-интерфейсы» и в строке «Конфигурация» должно стоять «Использовать DHCP». Нажать «Применить».

После этого по значку сети в нижнем углу экрана щёлкаем правой кнопкой мыши. Открывается окно «Активные сетевые соединения» (см рис. 1 Приложения 1).

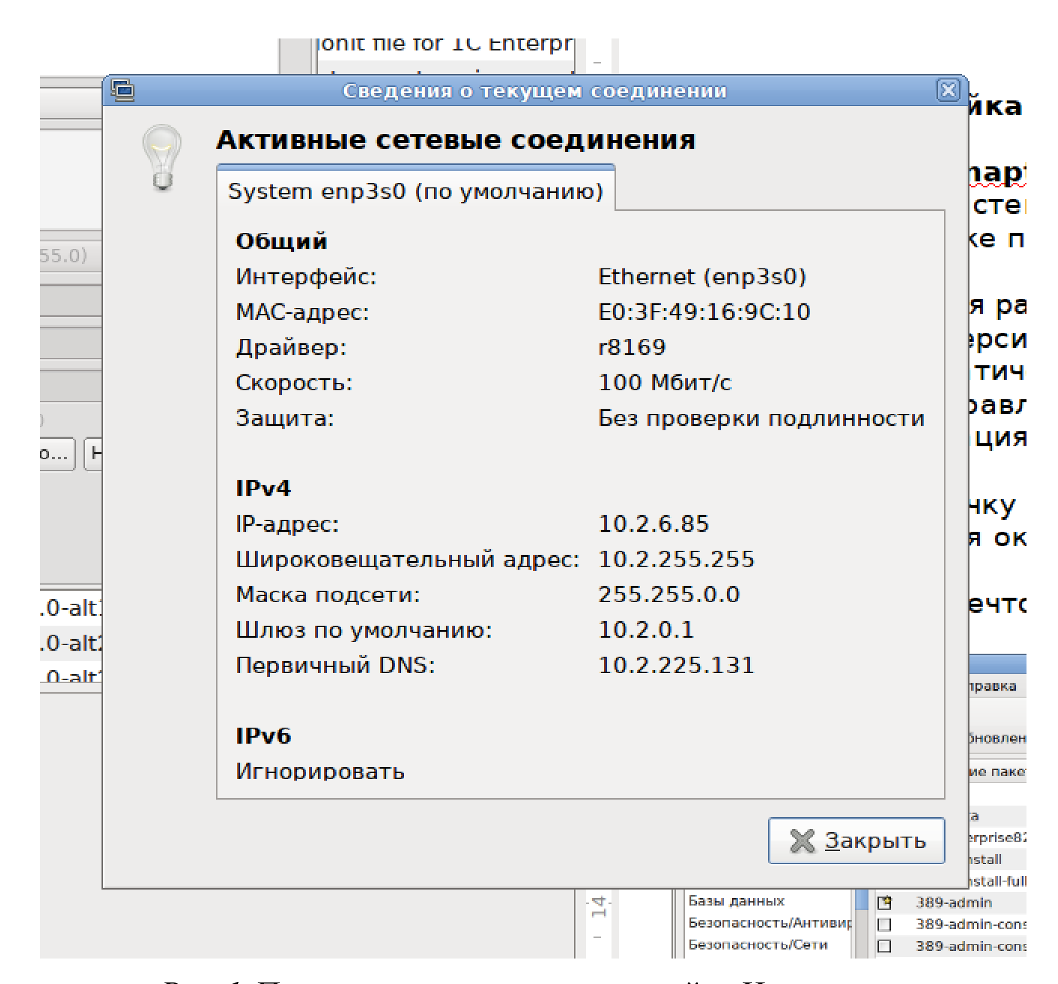



*Рис. 1.* Проверка правильности настройки Интернета

В этом окне мы видим настройки сетевого стека TCP/IP, в частности протокола IP версии 4 (IPv4). Если IP-адрес начинается с числа 10 (то есть, используется 10-я сетка адресов), то доступ в Интернет настроен. Альтернативно, доступность Интернета можно проверить в терминале с помощью команды ifconfig.

**Примечание.** Если интернет всё равно не работает, то проверьте правильность подключения к кабельной системе: в лаборатории две кабельных системы — нижние (жёлтые) розетки — с Интернетом, верхние (белые) — локальная кабельная система.

1.3. После запуска synaptic видим нечто такое:

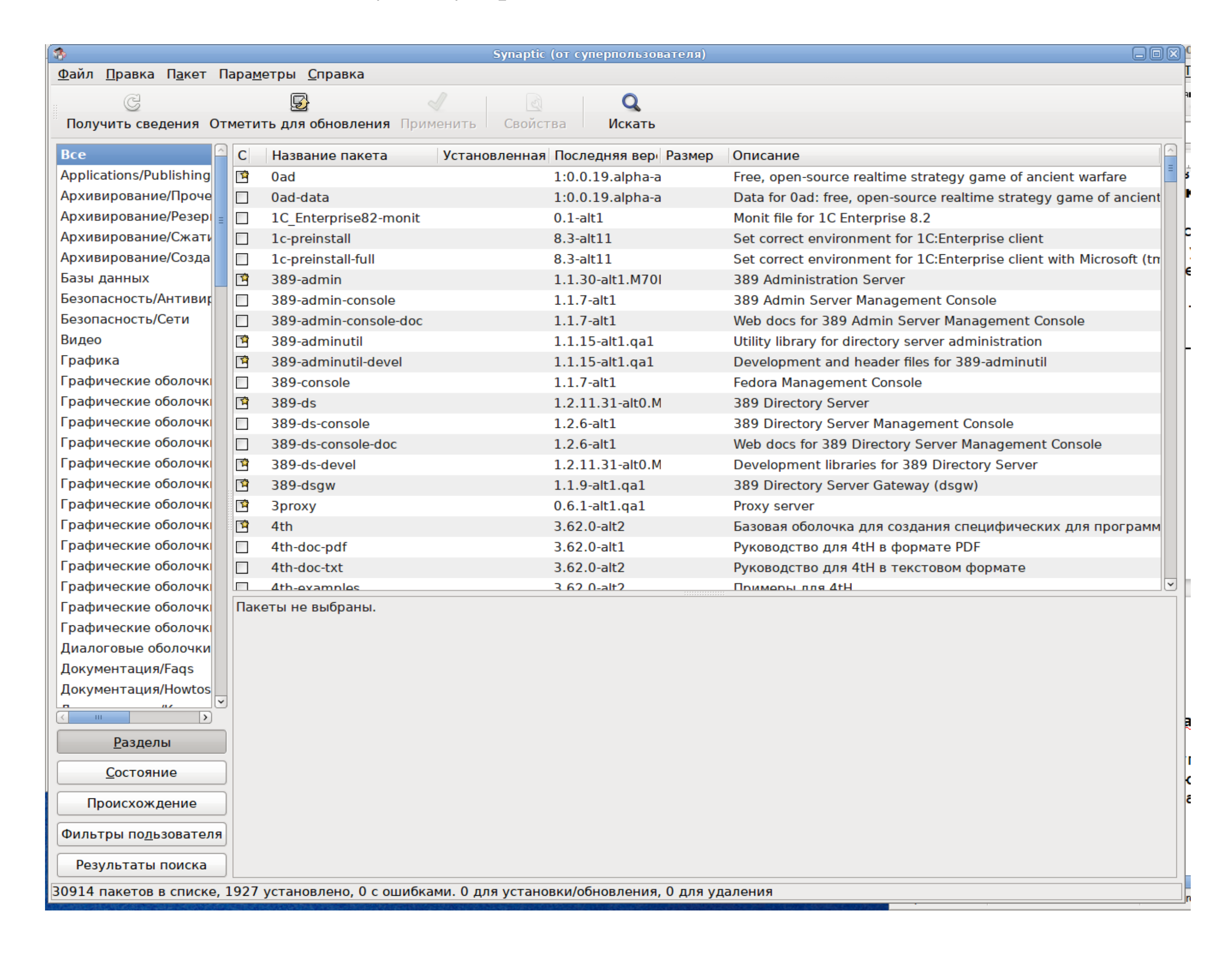



*Рис. 2.* Окно synaptic

1.4. В верхнем меню программы нажимаем «Параметры», затем «Репозитарии».

Открывается окно настроек репозитариев (см. рис. 3 Приложения 1).

Репозитарий — это база данных дополнительных программ, доступных в данном (который у вас установлен) дистрибутиве. Дополнительных в следующем смысле: когда вы ставите систему с CD/DVD/флешки/по_сети или из какого иного места, то вы ставите некоторый минимальный набор, определённый разработчиками дистрибутива. «Впихнуть» на носитель все-все доступные для данного дистрибутива программы, ессно невозможно почти всегда. Поэтому все остальные «дополнительные» программы раз-

мещаются на некотором ftp-сервере и к нему обеспечивается открытый доступ (обычно пользователю anonymous) из Интернета по протоколам ftp, http, sync. На этот ftp-сервер почти всегда можно зайти обычным браузером, увидеть файлы пакетов этих дополнительных программ и даже скачать их. А потом вручную установить. Но! В программах очень часто есть «зависимости». То есть, программы могут требовать предварительной установки некоторых других программ или библиотек. При ручной установке это всё также нужно отслеживать и предварительно в правильном порядке устанавливать. Программа synaptic этот процесс автоматизирует.

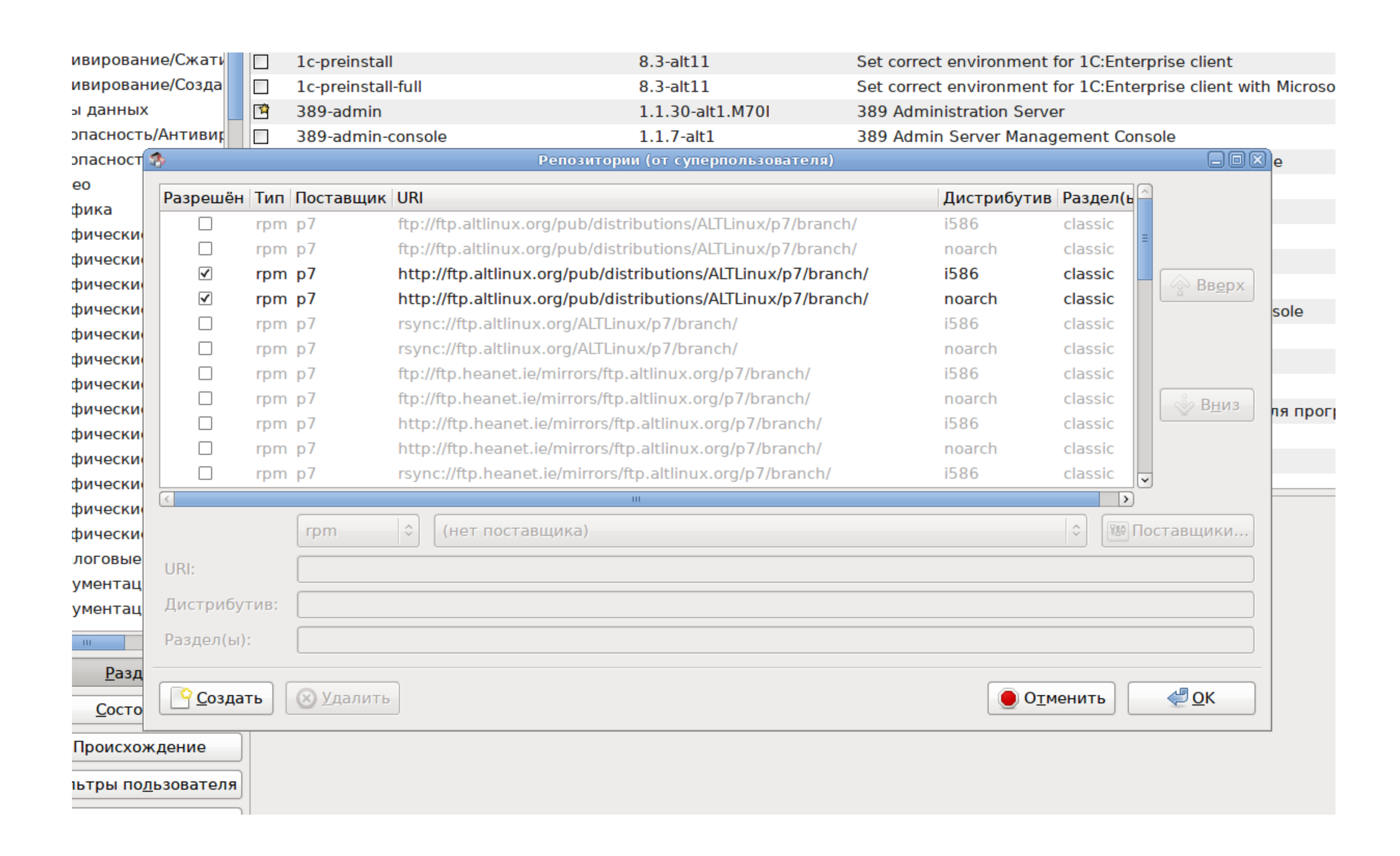



*Рис. 3.* Окно настройки репозитариев

1.5. В этом окне нужно поставить галочки напротив нужных репозитариев и только напротив них! Нигде больше галочек стоять не должно!

Пример: для дистрибутива altlinux KDesktop 7.05, которым в настоящее время пользуемся в лаборатории, галочки должны стоять так, как показано на рисунке 3, то есть, доступ к репозитариям по протоколу http, поскольку у нас доступ по другим протоколам «режется» прокси-сервером. Нажимаем «OK».

1.6. В верхнем меню программы нажимаем «Параметры», затем «Параметры». В открывшемся окне выбираем вкладку «Сеть» (см. рис. 4).

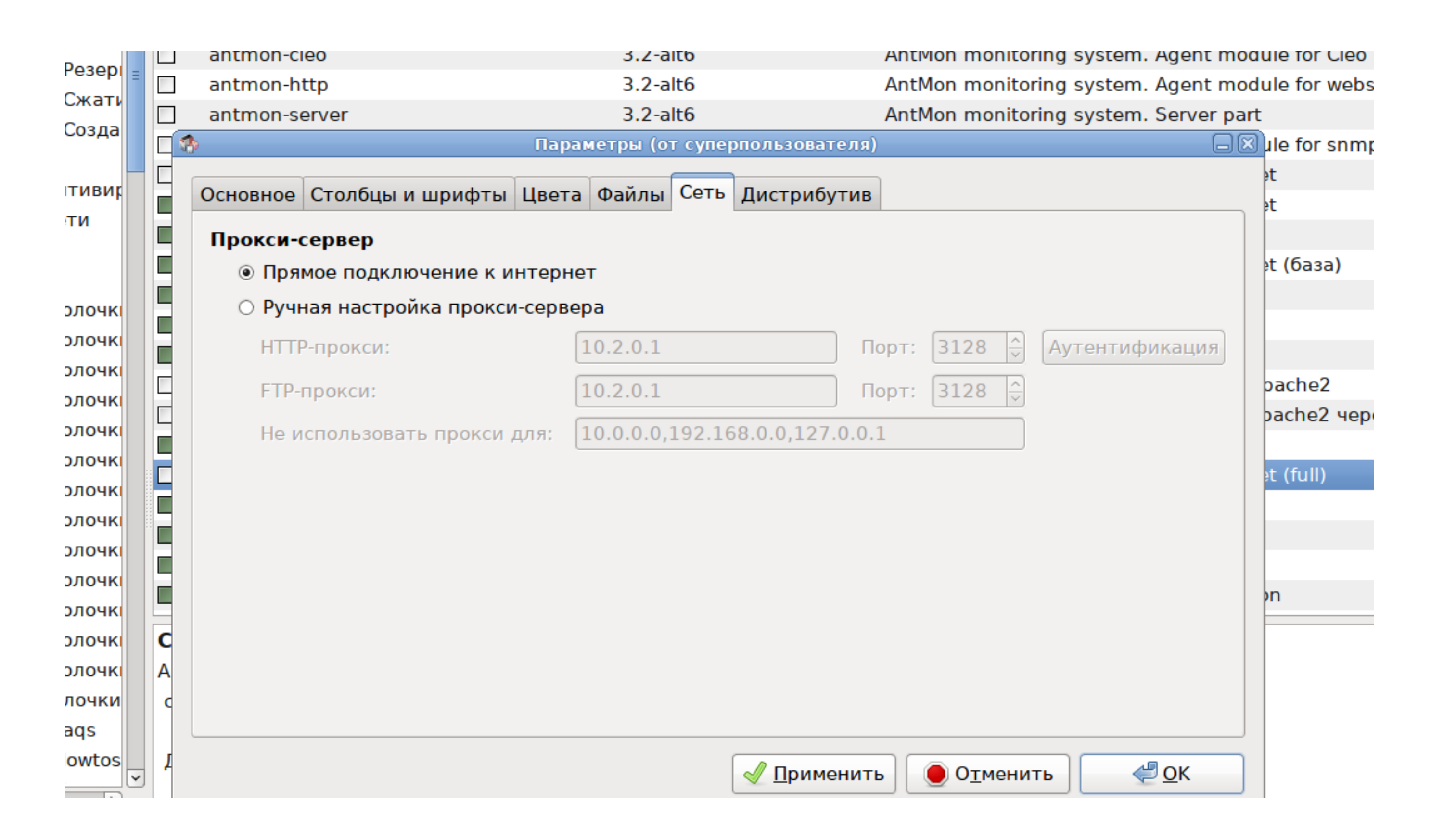



*Рис. 4.* Окно настойки подключения к сети

Смотрим настройки прокси-сервера. Должно быть установлено «Прямое подключение к интернет», поскольку в университете теперь прокси-сервер «прозрачный». Ранее требовалась «ручная настройка».

**Внимание!** В универе снова прокси-сервер «непрозрачный» и снова требуется ручная настройка, то есть, переключаемся на «Ручная настройка прокси-сервера» и вводим ip-адрес и порт proxy-сервера — на рисунке они указаны.

Нажимаем «ОК».

1.7. Теперь synaptic настроен. Можно начинать с ним работать.

**Внимание! Если вдруг дальнейшие действия не будут успешными, то выключите synaptic и запустите вновь.**

1.8. В верхнем меню нажимаем кнопку «Получить сведения». Открывается окно скачивания индексных файлов. Ждём завершения обновле-

ния индексных файлов. После того, как все 6 файлов скачаются и индексная база будет обновлена, это окно само закроется. После этого в нижнем левом углу окна synaptic появится информация о том, что доступно более 30 000 пакетов программ.

1.9. Работа с synaptic. Чтобы быстрее найти нужную программу можно

- либо нажать «Искать» и ввести начало названия программы,
- либо в левой части окна выбрать нужный раздел (щёлкнув по названию раздела) и в правой части окна поискать нужную программу в этом разделе.

Установленные программы помечены зелёными квадратиками, неустановленные — белыми.

Для установки программы щёлкаем правой кнопкой мыши по названию программы и выбираем «Отметить для установки». Можно так отметить несколько программ.

Затем вверху в меню нажимаем «Применить». Synaptic проверяет зависимости, организует скачивание и установку зависимых программ и в конце устанавливает выбранную вами.

Всё.

**Например, для установки web-сервера нужно искать «apache» и выбрать для установки пакет apache2-full.**

Для удаления программы щёлкаем правой кнопкой мыши по названию программы и выбираем «Отметить для полного удаления».

Затем вверху в меню нажимаем «Применить». Synaptic удаляет программу с компа. При этом удалятся зависимые библиотеки конкретно этой программы, остальные зависимые программы не удалятся.

Опять всё.

## ЛАБОРАТОРНАЯ РАБОТА № 2

**Тема:** Трансляция с веб-камеры в на свой сайт в Интернет в условиях динамического IP-адреса (фото и потоком).

Внимание! Вы сдавали лабораторную работу «Создание своего сайта для отчётов по лабораторным работам».

Именно на этот сайт добавить новую функцию «Трансляция с веб-камеры». Она должна быть оформлена как новый пункт в меню.

**Рекомендации и требования**

1. Требуется создать техническое решение, то есть:

- найти бесплатный hosting, на котором можно реализовать нужную функцию (hosting должен позволять установить на странице сайта плейер);
- оформить нужным образом страницу сайта;
- настроить домашний комп на работу с web-камерой;
- сконфигурить ПО на передачу видеотрафика с домашнего компа в плейер на страничке своего сайта, а плейер, соответственно, на приём видеотрафика с вашего домашнего компа.

2. Поскольку провайдер Интернета предоставляет клиентам, как правило, «серые» адреса (из приватных сеток), то, очевидно, необходимо задействовать динамический DNS и правильно настроить свой маршрутизатор.

**См. приложенный пример реализации.**

3. Рекомендуется показать «жизнь за окном».

**Предупреждение: Есть сервисы в Интернете ivideon и аналогичные — очень удобные, можете их попробовать. Но сдавать работу всё равно будете в полном объёме со своими настройками dDNS и прочего. См. ниже пример. Ivideon и аналоги знать надо, но работу сдавать в полном объёме.**

4. **Можно (дополнительно!!) настроить трансляцию на сайт со своего смартфона** — см. пример приёма трафика со смартфона плейером сайта — пункт 4 приложения.

**Порядок сдачи лабораторной**

Продемонстрировать в лаборатории 326:

- подключиться браузером к своему сайту и показать, что видеотрансляция работает.

В отчёте о выполнении данной работы должны быть:

- задание;
- описание использованного ПО;
- описание конфигурации — конфигурационные файлы со скринами.

Срок сдачи лабораторной — до ДД.ММ.ГГ.

Проблемы:

https://help-wifi.com/poleznoe-i-interesnoe/ddns-dinamicheskij-dns-na-routere-chto-eto-kak-rabotaet-i-kak-polzovatsya/

https://wifika.ru/dyndns-ddns.html

**ПРИЛОЖЕНИЕ 1 к лабораторной №2**

**Один из возможных вариантов. Пример решения:**

«Трансляция видео с веб-камеры в Интернет»

**1. Постановка задачи**

Используя возможности медиаплеера VLC организовать трансляцию изображения с веб-камеры в Интернет (в условиях динамического ip-адреса).

**2. Ход работы**

В ходе работы выполняются следующие действия.

**2.1. Подготовительные действия**

2.1.1. На машине, с которой будет вестись трансляция, произведена установка VLC.

2.1.2. Определено имя веб-камеры, как устройства в системе. Для этого до и после подключения камеры к USB-порту выполнена команда:

$ ls /dev/video*

После сравнения результатов выполнения сделан вывод о том, что веб-камера инсталлировалась в системе под именем /dev/video2.

**2.2. Обеспечение доступности локальной машины «извне»:**

2.2.1. Ввиду того, что провайдер предоставляет для выхода в Интернет динамический ip-адрес (меняющийся каждый раз при установке VPN-соединения), принято решение использовать сервис динамического DNS No-IP (http://www.noip.com/).

После регистрации учетной записи и выбора имени хоста (fedork.ddns.net) полученные данные были внесены в соответствующий раздел веб-интерфейса маршрутизатора ZyXEL Keenetic Giga II, находящегося в структуре сети между локальной машиной и выходом в Интернет (см. рис. 1).

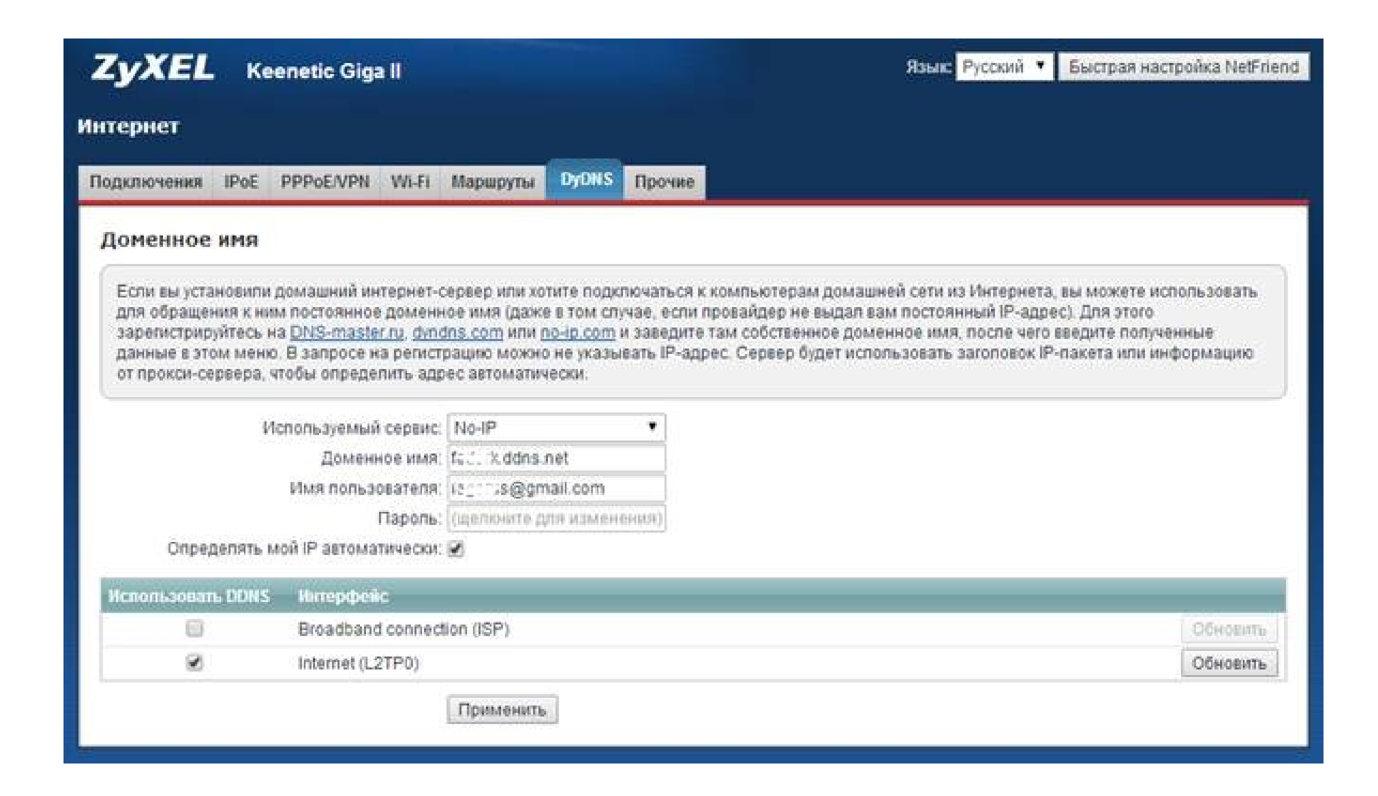



*Рис. 1.* Назначение имени компа

2.2.2. Произведено назначение постоянного внутрисетевого ip-адреса локальной машине (см. рис. 2).

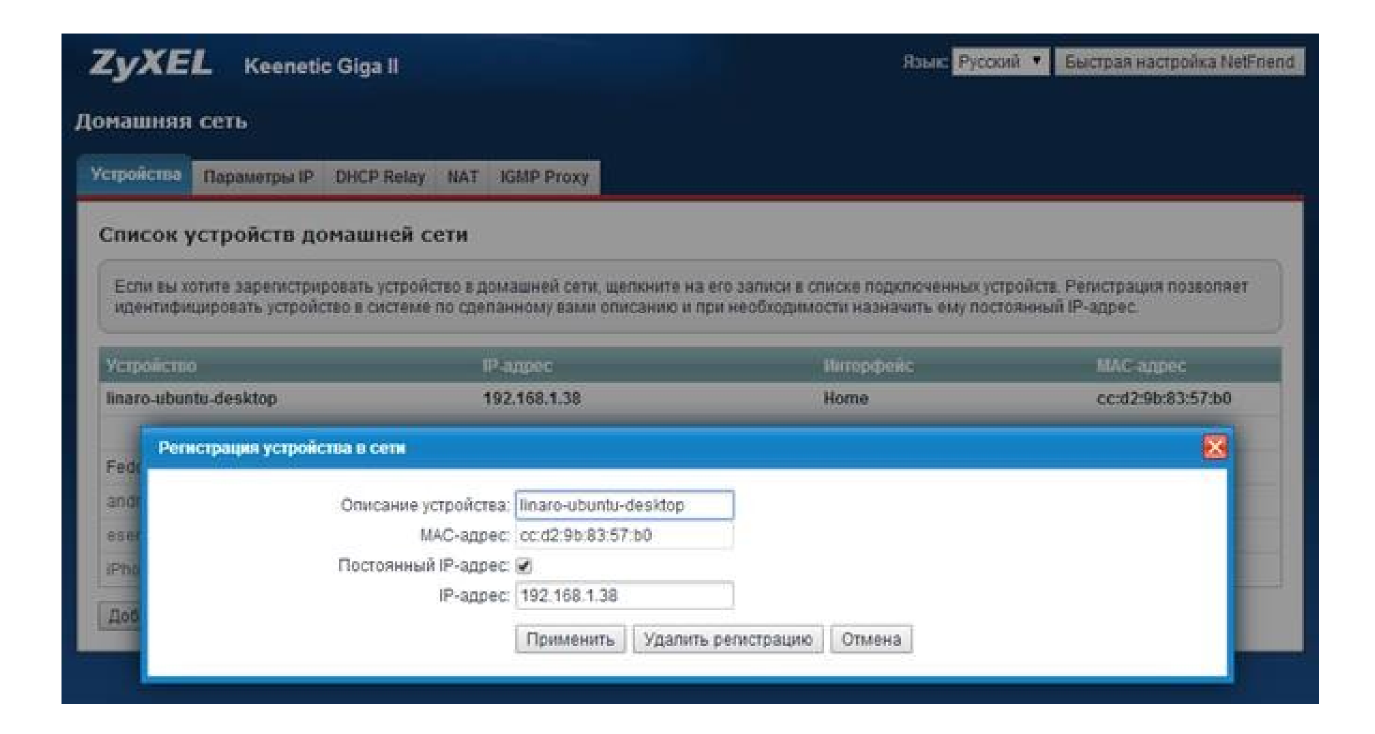



*Рис. 2.* Назначение ip-адреса

2.2.3. Для прохождения запросов из сети Интернет к локальной машине задано правило трансляции адресов (NAT) (см. рис. 3).

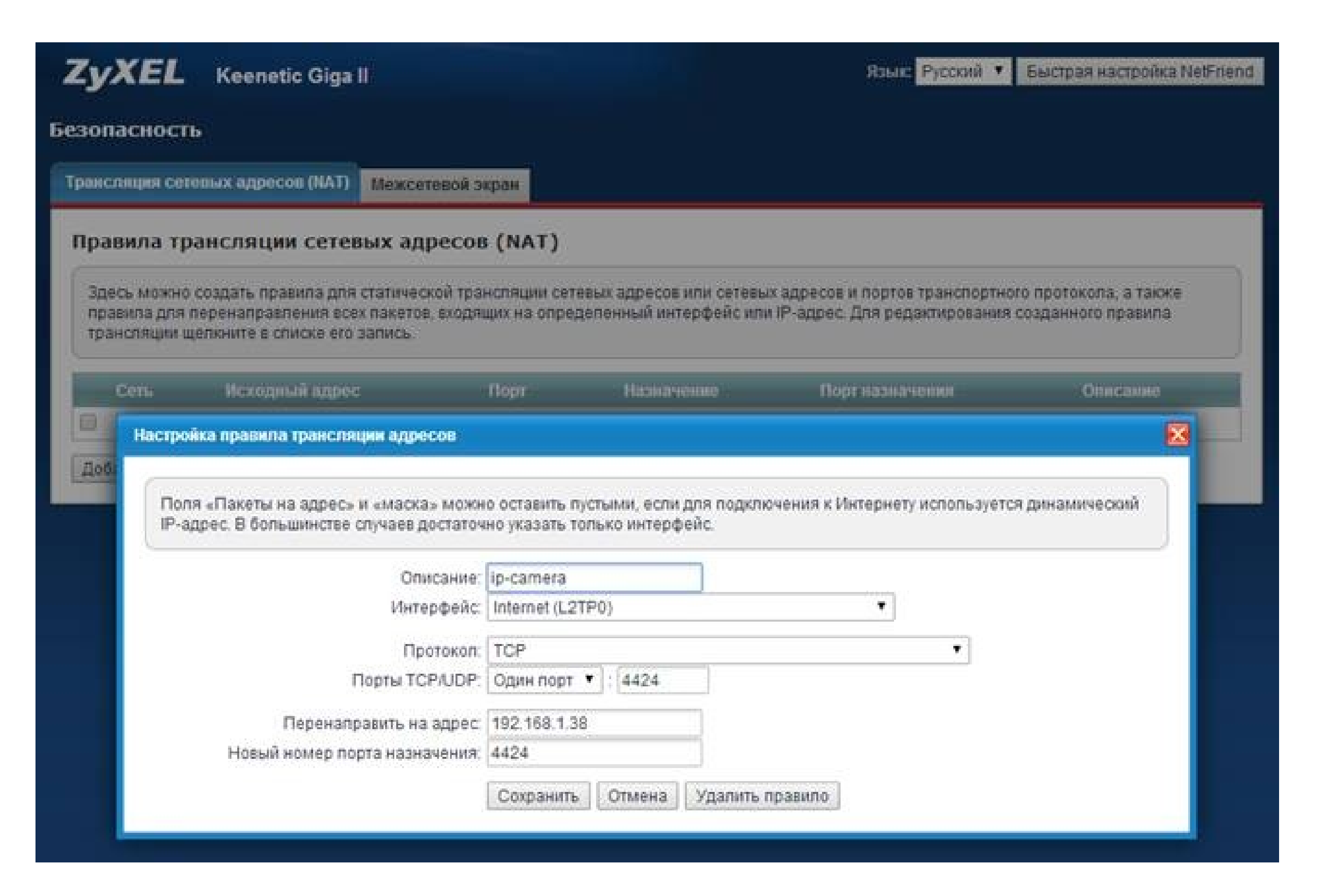



*Рис. 3.* Настройка NAT в маршрутизаторе

**3. Запуск потоковой передачи видео:**

3.1. Запущена потоковая передача изображения с камеры средствами VLC.

$ cvlc v4l2:///dev/video2 :sout="#transcode{vcodec=FLV1,vb=2000, fps=25,acodec=none}:standard{access=http{mime=video/x-flv}, mux=ffmpeg{mux=flv},dst=:4424/stream.flv}"

Указаны битрейт видеопотока, кадровая частота и видеокодек. В качестве видеокодека и метода инкапсулирования выбран формат FLV. Поток в данном случае передается с помощью встроенного в VLC HTTP-сервера через заданный порт.

3.2. Указанная выше команда добавлена в список автозагрузки системы.

**4. Демонстрация потокового видео на веб-странице:**

4.1. В качестве плеера, встраиваемого на веб-страницу и воспроизводящего FLV-поток, выбран Flowplayer.

4.2. Страница со сконфигурированным плеером (см. листинг 1) загружена на сервер и доступна по адресу http://lab.XXXork.com/os/lab2.

```
<!doctype html>
<head>
 <link rel="stylesheet" href="skin/minimalist.css">
 <script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
 <script src="flowplayer.min.js"></script>
</head>
<body>
 <div class="flowplayer" data-swf="flowplayer.swf">
  <video>
   <source type="video/flash" src="http://XXXork.ddns.net:4424/stream.flv">
  </video>
 </div>
</body>
</html>
```

*Рис. 4.* Листинг странички

**4. Примеры приёма трафика со смартфона плейером сайта.**

```
<tr>
<td colspan = '2'>
 <iframe src="https://player.twitch.tv/?channel=mulyan" frameborder="0" allowfullscreen="true" scrolling="no" height="378" width="620"></iframe>
 <a href="https://www.twitch.tv/mulyan?tt_content=text_link&tt_medium=live_embed" style="padding:2px 0px 4px; display:block; width:345px; font-weight:normal; font-size:10px; text-decoration:underline;"></a>
 <p style = "text-align: bottom;" style = "text-align: center;"> </p>
</td>
</tr>
```

или так:

```
<div class="layer">
  <center> <h1><b>My cam</b></h1> <hr> </center>
  <p><b>Тема:</b>Трансляция изображения с камеры в Интернет</p>
  <iframe src="http://openfuturecam.ddns.net:8080/" frameborder="0" allowfullscreen="true" scrolling="no" height="378" width="620"></iframe>
<a href="http://openfuturecam.ddns.net:8080/"
    style="padding:2px 0px 4px; display:block; width:345px; font-weight:normal; font-size:10px; text-decoration:underline;"></a>
</div>
```

или так:

```
<div class="layer">
  <center> <h1><b>Отчет 2</b></h1> <hr> </center>
  <p><b>Тема:</b>Трансляция изображения с камеры в Интернет</p>
  <iframe src="https://player.twitch.tv/?channel=alexk4321"
frameborder="0" allowfullscreen="true" scrolling="no" height="378"
width="620"></iframe>
  <a
href="https://www.twitch.tv/alexk4321?tt_content=text_link&tt_medium=live_embed" style="padding:2px 0px 4px; display:block; width:345px; font-weight:normal; font-size:10px; text-decoration:underline;"></a>
</div>
```

## ЛАБОРАТОРНАЯ РАБОТА № 3

**Тема:** Установка и конфигурирование файлового сервера рабочей группы (ftp+nfs+samba+video+web).

**Рекомендации и требования**

1. Для выполнения лабораторной работы необходимо правильно настроить сеть.

2. Подключение к удалённому компьютеру осуществлять по hostname или по полному имени.

3. Для выполнения лабораторной работы доустановить необходимые пакеты, если это необходимо.

4. Доступ к расшаренным ресурсам разрешать только для определённых host'ов — указывать точно. Кроме сервисов video и web — для этих сервисов доступ не ограничивать. Определить в конфигурациях — кто имеет право пользоваться файловым сервером.

**Порядок сдачи лабораторной**

Установить в лаборатории 326 файловый сервер. Настроить. Продемонстрировать доступ с разрешённых компьютеров.

Продемонстрировать отсутствие доступа с других компьютеров.

1. ftp:

1.1. Показать подключение зарегистрированным пользователем. В этом случае доступ должен быть в домашний каталог по чтению/записи, в общие каталоги: pub — по чтению, в incoming — чтение и запись.

Внимание: настройки не должны позволять пользователю выходить за пределы указанных каталогов, иначе говоря, ничего кроме не должно быть видно.

1.2. Показать подключение анонимным пользователем. В этом случае доступ должен быть только в общие каталоги: pub — по чтению, в incoming — чтение и запись.

1.3. Для доступа к сервису использовать консольный клиент lftp (установлен по умолчанию). Смотри man lftp.

1.4. Дополнительно могут использоваться ftp-клиенты mc, браузера, графические клиенты.

2. nfs:

2.1. Монтирование только вручную. Ни в коем случае не вставлять монтирование в fstab. Объяснить, почему нельзя.

3. samba:

3.1. Показать видимость сервера в сетевом окружении Винды и доступ к серверу с Виндовой машины.

4. video:

4.1. Показать доступ с помощью плейера, Например, vlc.

5. web:

5.1. Показать доступ с помощью браузера. Контент web-сервера — ваш сайт. Смотри соответствующую лабу.

В отчёте о выполнении данной лабораторной работы должны быть:

- задание;
- описание настройки сети;
- описание конфигурирования файлового сервера (ftp+nfs+samba+video+web);
- описание выполненной работы со скринами.

Срок сдачи лабораторной — до ДД.ММ.ГГ.

**Дополнительная информация**

Смотреть на altlinux.org в разделе «Руководства».

Например:

1. ftp

https://docs.altlinux.org/ru-RU/archive/4.0/html-single/server/vsftpd/index.html

http://tekct.ru/altlinux/8_5__dostup_k_server.htm

http://www.linuxcenter.ru/lib/articles/soft/vsftpd_setup.phtml

Также приложено руководство администратора.

2. nfs

https://www.altlinux.org/NFS]

http://wiki.sisyphus.ru/net/nfs

http://www.botik.ru/rented/rldp/www/ldp/nag-20/mountd.htm

http://nfs.sourceforge.net/

http://www.intuit.ru/department/os/adminsolaris/8/

http://sky.inp.nsk.su/~bolkhov/teach/inpunix/setup_nfs.ru.html
http://skif.bas-net.by/bsuir/base/node87.html
http://www.freebsd.org/doc/ru_RU.KOI8-R/books/handbook/network-nfs.html

3. samba
https://www.altlinux.org/Samba
http://tekct.ru/altlinux/8_8__ustanovka_pervi.htm
http://www.samba-doc.ru/adaptation/adap03.html

**Приложение 1 к лабораторной работе №3**

**О настройке ftp и его использовании**

Настройки ограничений доступа к ftp-серверу могут быть реализованы двояко:

1. Ограничения на уровне пользователей и каталогов — в конфигурационном файле демона сервера. Смотри соответствующие руководства.

2. Ограничения на уровне сеток и компов могут быть дополнительно реализованы с помощью настроек суперсервера xinetd, например для сервера wsftpd в файле /etc/xinetd.d/wsftpd могут быть такие настройки:

```
# default: off
# description: The vsftpd FTP server.
service ftp
{
  disable = no # включает службу
  socket_type = stream
  protocol = tcp
  wait = no
  user = root
  nice = 10
  rlimit_as = 16M # устанавливает лимит адресного пространства
  server = /usr/sbin/vsftpd # путь к исполняемому файлу
  only_from = 192.168.0.0 # предоставляем доступ из всей подсети 192.168.0
  only_from = 207.46.197.100 207.46.197.101 # доступ с указанных адресов
```

# only_from = 0.0.0.0 # неограниченный по адресам доступ

access_times = 2:00-9:00 12:00-24:00 # время, когда возможен доступ

}

Для получения дополнительной информации по использованию xinetd смотрите

man xinetd

man xinetd.conf.

**Замечание: пользователь может работать на разных компьютерах. Если такое наблюдается, то для организации и контроля доступа для этого пользователя может использоваться только первый способ — с помощью конфигурационного файла демона сервиса.**

## Список литературы

1. Чичев А. А., Чекал Е. Г. Операционные системы. Часть 1. Работа с операционной системо : учебное пособие. — Ульяновск : УлГУ, 2015.

2. Чичев А. А., Чекал Е. Г. Операционные системы. Часть 4. Фонд оценочных средств : учебное пособие. — Ульяновск : УлГУ, 2019.

3. Чичев А. А., Чекал Е. Г. Администрирование информационных систем. Часть 1. Общие вопросы : учебное пособие. — Ульяновск : УлГУ, 2018. — 156 с.

4. Чекал Е. Г., Чичев А. А. Надёжность информационных систем. Часть 1 : учебное пособие. — Ульяновск : УлГУ, 2012.

5. Чичев А. А., Чекал Е. Г. Архитектура инфокоммуникационных устройств. Часть 2. Лабораторные работы : учебное пособие. – Ульяновск : УлГУ, 2017. — 156 с.

6. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. — 2-е изд. — СПб. : Питер, 2009. — 669 с. : ил.

7. Олифер В., Олифер Н. Компьютерные сети. Принципы, технологии, протоколы : учебник. — 5-е изд. — СПб. : Питер, 2016. — 992 с. : ил. — (Серия «Учебник для вузов»). — ISBN: 9785496019675.

8. Таненбаум Э., Уэзеролл Д. Компьютерные сети. — 5-е изд. — СПб. : Питер, 2016.

9. Таненбаум Э., Вудхалл А. Операционные системы. Разработка и реализация. — 3-е изд. — СПб. : Питер, 2007. — 704 с. : ил.

10. Куроуз Д., Росс К. Компьютерные сети: Нисходящий подход. — 6-е изд. — М. : Изд-во «Э», 2016. — 912 с. : ил. — ISBN 978-5-699-78090-7.

11. Куроуз Д., Росс К. Компьютерные сети. Настольная книга системного администратора. — 6-е изд. — М. : Эксмо, 2016 — 912 с. : ил. — ISBN 978-5-699-94358-6, 0132856204.

12. Робачевский А. Интернет изнутри. Экосистема глобальной сети. 2-е изд. — М. : Альпина Паблишер, 2015. — 223 с. : ил. — ISBN 978-5-9614-4803-0, 978-5-9614-5882-4.

13. Уэнделл О. Официальное руководство Cisco по подготовке к сертификационным экзаменам CCNA ICND2 200-101: маршрутизация и коммутация, акад / пер. с англ. — М. : ООО «И. Д. Вильямс», 2015. — 736 с. : ил. — ISBN 978-5-8459-1907-6.

14. Хант К. TCP/IP. Сетевое администрирование. — 3-е изд. / пер. с англ. — СПб. : Символ-Плюс, 2007. — 816 с. : ил. — ISBN-10: 5-93286-056-1, ISBN-13: 978-5-93286-056-4.

15. Ногл М. TCP/IP : иллюстрированный учебник — М. : ДМК Пресс, 2001. — 480 с. : ил. — ISBN 5-94074-044-8.

16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. : ил. — (Сер. «Классика computer science»). — ISBN 978-5-496-01395-6.

17. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е изд. — СПб. : Питер, 2018. — 816 с. : ил. — ISBN 978-5-496-00337-7.

18. URL: //top500.org (дата обращения: 07.12.19).

19. Чичев А.А., Чекал Е.Г. Стилевые конфликты и оценка стиля программирования : сб. тр. XI междунар. конф. «Информационные технологии в образовании». Ч. 2. — М. : МИФИ, 2001. — С. 151-154.

20. Чичев А.А., Чекал Е.Г. Подготовка ИТ-специалистов в университете // Ученые записки УлГУ. Сер. «Математика и информационные технологии». — Вып. 1(4). — Ульяновск : УлГУ, 2012. — 226 с. — С. 278-286.

21. Айзерман М. А., Гусев Л. А., Розоноэр Л. И., Смирнова И. М., Таль А. А. Логика, автоматы, алгоритмы. — М., 1963.

22. Виноградов И.М. Математическая энциклопедия. — М. : Советская энциклопедия, 1977-1985.

23. Кудряшов А.В., Шлеменкова Е.О., Чичев А.А., Чекал Е.Г. Архитектура распределённой гуманоидной роботехнической системы «АЛКЕТОН-УлГУ» // Ученые записки УлГУ. Сер. «Математика и информационные технологии» [Электр. журн.]. — Ульяновск : УлГУ, 2019. — № 2. — С. 54-68.

24. Кудряшов А.В., Шлеменкова Е.О., Чичев А.А., Чекал Е.Г. Алгоритмы функционирования распределенной гуманоидной роботехнической системы «АЛКЕТОН-УлГУ» // Ученые записки УлГУ. Сер. «Математика и информационные технологии» [Электр. журн.]. — Ульяновск : УлГУ, 2020. — № 1.

25. Ахметов Д.М., Чичев А.А. Реализация вычислительного кластера в УлГУ // Ученые записки УлГУ. Сер. «Математика и информационные технологии». — Ульяновск : УлГУ, 2012. — Вып. 1(4).

26. Ахметов Д.М., Чичев А.А. Организация удаленного доступа к кластеру // Ученые записки УлГУ. Сер. «Математика и информационные технологии». — Ульяновск : УлГУ, 2012. — Вып. 1(4).

27. Таненбаум Э., ван Стеен М. Распределённые системы. Принципы и парадигмы. — СПб. : Питер, 2003. — 877 с. : ил. — (Сер. «Классика computer-science»). — ISBN 5-272-00053-6.

28. Учебники: http://www.ripe.net/support/training/material.

*Учебное издание*

***Чичев А.А., Чекал Е.Г.***

# СЕТЕВОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

*Учебное пособие*

Директор Издательского центра *Т.В. Максимова*
Подготовка оригинал-макета *М.А. Водениной*

Издается в авторской редакции

Подписано в печать 22.12.2020.
Формат 60×84/16. Усл. печ. л. 9,8.
Тираж 100 экз. Заказ № 138/

Оригинал-макет подготовлен и тираж отпечатан в Издательском центре
Ульяновского государственного университета
432017, г. Ульяновск, ул. Л. Толстого, 42